DATA VALIDATION AND NORMALIZING FOR A PERSONNEL DATA BASE BUILT ON THE RELATIONAL APPROACH

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by

AJOY KUMAR MUKHERJEE

to the

COMPUTER SCIENCES GROUP
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
JULY 1977

2.7.77

CERTIFICATE

CERTIFIED that the work entitled, 'DATA VALIDATION AND NORMALIZING FOR A PERSONNEL DATA BASE BUILT ON THE RELATIONAL APPROACH' is carried out under my supervision by Sri Ajoy K. Mukherjee and it has not been submitted elsewhere for a degree.

Kanpur
July 18, 1977

R. Sankar
Dr. R. Sankar
Professor of Computer Science
Indian Institute of Technology
Kanpur

ACKNOWLEDGEMENT

I wish to take this opportunity to record my deepest gratitude to Dr. R. Sankar in whom I found a most persevering paternal figure, who has been unfailingly at my side whenever I have failed and been a constant source of inspiration to me without which I would have found this work difficult.

I also wish to thank Major R.K. Sharma and Lt. Col. R.K. Bagga, without whose help I could not have got through the jungle of army nomenclatures I had to deal with in structuring the data base model. Thanks are due in a large volume to Deepak K. Ghanekar who assisted me through out the project. I am also grateful to the above three for the lively discussions we had on the topic and for all that I was able to learn through it. I am grateful to all my friends who have made my stay here pleasurable.

I also wish to thank Mr. H.K. Nathani for his excellent and elegant typing and the staff of the Computer Centre for all the cooperation I have received from them.

- Ajoy K. Mukherjee

Kanpur
July 18, 1977

CONTENTS

ABSTRACT

This project consists of a part of the building of a DBMS based on the relational model. The data base chosen was the personnel data base of the Army Officers. In this part, the data base structure was first organized in the third normal form representation of relations. The data was then validated and placed on the disk according to that structure. The system was implemented on TDC-316. This project was part of a larger data base project. The other parts developed were:

[a] Building of the Data Base

[b] Information Retrieval from the Data Base.

All the systems are written in the Basic Assembly Language of the TDC-316.

1. INTRODUCTION

In this project we are going to consider data validation in data base system. The data base structure was built on the relational approach.

Since the first publication of Codd's work [1] and his later publications [2,3,4], the area of relational approach to data base management systems has been a hot bed of beehive like activities. This approach has been now considered from practically most of the theoretical angles and a large volume of work carried out in all these directions. Date [5] and Martin [6] provide excellent treatment of these topics. Especially Date's treatment provides the best reference and utilised in this present work. The relational model has been proved theoretically to be superior to the other two approaches of hierarchical and network or DBTG. But in one respect viz., implementation it is sadly lacking. While there are some good major implementations of the hierarchical type (IMS of IBM is a classic and successful example) and several on the DBTG approach, there are still no major implementations and very few minor ones based on the relational approach. Some of these are mentioned by Chamberlin [15].

This project was taken up with this specific aim in mind. We wanted to implement a data base management system based on the relational approach. Our desire was to probe behind the facade of theoretical impregnability and find

for ourselves the difficulties faced in its practical implementation. Since a major project of this sort could not be handled in such a short span of time by a single person, it was split into several sections, to be attempted for implementation by different persons.

This section of the project, as mentioned earlier, deals with the data validation and determining the structure of the data base. Data validation is a very important part of a DBMS, as we shall demonstrate presently. To do so, we would perhaps do well to go back a little and describe how data base management systems came about in the first place.

1-2. DEVELOPMENT OF DATA BASE TECHNOLOGY AND ITS RELATION TO DATA VALIDATION

Sibley [7] has described how and why this development came about. When computer age began the data processing continued in the same manner as in the previous eras, i.e., each program had its own set of data and was owner of this data. Any other user found it difficult to obtain, integrate, or transform the 'available' data for use in another program. Thus, every new need for data involved writing a new program to obtain the data before it could be processed by yet another program, and even then it was difficult to do so - the data formats were 'locked' in the original programs, and sometimes the original object codes had been lost. This also led to a large amount of duplication of data and effort and

also the programs were completely data-dependent. The essential unavailability of otherwise transferrable data gave rise to the question:

'Why not integrate the data?'

This led to the thought that integration, were it possible, could be achieved by defining the data format, storing it as a 'data definition' and allowing general-purpose 'database management' software to access it. This gave rise to the concept of the generalized data-base management system. Next came up the question: 'Can we access this data through our current computer languages?'

And another: 'Why not allow a higher level language for adhoc use of the data base?'

Though inefficient, the first prototype systems appeared very successful to users, and commercial systems started to appear. Then the first problems arose.

The industry congratulated itself on reducing data redundancy and improving its availability, but it also introduced the potential for disaster. The first problem with integration arises because the data base is now more vulnerable to destruction through machine malfunction, personal error or deliberate human tampering. The loss of 'quality' in a data base (including total desctruction) by any of these means may be considered a threat to the very existence of the organisation, because data is one of its more valuable assets.

'Integrity' techniques are therefore a necessity.

We may look at it from another point of view. Lott [8] says that, in data processing, we should remind ourselves of the saying that 'anything worth doing is worth doing right.' Our operations are costing us so much to perform, and we hope they are being relied upon by so many people, that we must satisfy ourselves that we are operating within reasonable limits of correctness. Manual systems can have things go wrong intentionally and unintentionally - the more we mechanize and convert to electronics, the faster certain portions are performed and the faster incorrect output can be developed if proper controls are not employed to catch and to prevent such errors. But we must be cautious to have the right mixture of controls; on the one hand we want to catch (and preferably prevent) as many errors as we can. And on the other hand we want to keep the costs of control to a reasonable figure. We cannot afford cent per cent control. Thus we arrive at the conclusion that one of the primary and yet most important requirement for a DBMS is the process of data validation. Not only programs, but also the data they operate on must be made as nearly accurate as human frailty and economic conditions will allow. Unless the data, on which the required operations of the DBMS are to be performed, is reasonably pure, the value of the DBMS is greatly reduced. This is clearly spelt out by the acronym GIGO (standing for garbage in, garbage out) meaning that if

the data you feed into the system is trash, all you can expect out of the system is trash only.

But possibly the most neglected objective of DBMS is the maintenance of quality. Problems relating to the quality of data and the integrity of systems and data go hand in hand.

1-3. A SURVEY OF THE SOURCES OF ERROR AND THE METHODS USED TO CONTROL THESE

The data used in a DBMS can be erroneous due to several reasons. Errors can creep into the data from various sources. Some of these are mentioned in Date [5] and will be outlined here with their methods of prevention wherever possible.

(1) Error at source, i.e., where the data has been collected. This cannot be directly verified, so either it must be double-checked at the source or some such fool proof method used (generally non-existant) for form design, data collection and coding, etc., that the errors are at once detected.

(2) Error after entering the machine, due to hardware failure at any point in the system may be a source of error. For example, the channel used for data transfer from the memory to the disk packs or the tape units may have one of the bit transfer lines out of order at any particular instant of time so that an error in that position may occur in some part of the stored data. Some of these errors are dealt with by the inbuilt mechanisms of the system itself, as the parity checking mechanism in tape drives, etc. We may mention in this regard,

the mechanism of error detection built into the disk drive controller of the TDC-316. It computes a Cyclic Redundancy Check sum Code (CRCC) for every sector of data that it stores on the disk pack while the writing operation is going on. Next, when it is reading that sector of data, it recomputes the CRCC and matches it to its previously calculated value stored at the end of the sector, any discrepancy causing error to be indicated and the read operation aborted.

(3) Human errors play a significant part. The error during the collection of data has already been mentioned. Other human errors may be on the part of a computer operator, a keypunch operator or a terminal user. A typical error due to carelessness of a computer operator, that frequently plagues our installation here, is that the tape drives are not cleaned properly before mounting the tapes. Then, if some impurity such as dust particles were present the data stored may be corrupted. This may again be taken care of by the system, e.g., the IBM 7044 here gives IOBS errors in such cases generally. The operator may carelessly not repeat the data cards which have given read-check at the reader, so that the data on these cards is not loaded into the system at all and if the data was being loaded according to some sequence, that sequence may be totally lost. This is very difficult to detect, unless a total dump of the data is taken and the sequence meticulously checked, involving a lot of effort and time.

Human errors in keypunching are very much prevalent. These may be of several types, e.g., transcription errors in which the digits are punched wrongly, transposition errors in which digit positions get interchanged, shift errors by which a number like 24540 may be punched as 245400 (left shift error) or as 2454 (right shift error), etc. An example of the last type of error can very easily occur during formatted FORTRAN I/O. The most prevalent amongst the keypunching errors are the single transcription (one digit punched wrongly) and single transposition (a pair of digits, generally adjacent, getting interchanged). These two together account for nearly 90 percent of the keypunching errors.

As Emery [14] has described, these errors may generally be checked at source. One process suggested by him is to prepare punched cards and punched paper tapes twice, inspite of the added expenditure, and then compare them to eliminate errors. Another process, in common practice, is to use the dual process of punching and verification. A surer method of course is to use some form of error detecting code, like the modulus 11 code, when writing down the data itself. But these have two serious drawbacks. One is that it involves a large amount of computation to find the check digits themselves. The other is that it again involves human computation and so introduces new chances of error. But when taking the output from the computer, to be used subsequently as input at some later stage, these codes may be utilized.

Error possibilities exist even in the case of terminal users, but their probability of occurrence is very low and the possibility of early detection and correction is very high, so these need not be considered.

(4) Programming errors in the DBMS or the underlying operating systems may be another source of error. We can take a simplified example. Let there exist in the DBMS a program that maps the logical record by its primary key to some physical record position on the disk. Let us assume that the program takes the help of a random number generator in achieving this mapping. Now if there are bugs in this random number generator (though such a case should not normally occur), then it may very well map the record onto some position on the disk while writing and then when retrieval is being attempted, it might read some other record, even may be record of a totally different type which may then be corrupted through manipulation. The operating system may also, due to some internal bug, fail to attach the proper physical record sequence of track, surface, etc., for a particular logical record asked for, thus giving rise to error.

(5) Programming errors in the data base application programs constitute another source of error. These may very well cause data corruption. A program may, while manipulating with the data, place the fields in some wrong sequence or it may add the wrong things to the wrong fields or it may even

access wrong portion of the data base by giving wrong identifiers for the files, records, etc. The last error is generally prevented by adding security locks to the different portions of the data base.

(6) The last source of error mentioned by Date is valid for a shared multiple-user system. In such a system, it is always possible that two or more users may have access to the data base at the same instant of time, and that the same record may be updated by different users at the same time differently. For example, in a real time airlines reservation system, say the number of seats available on a particular flight on a particular day is five, at a particular instant, and two reservation clerks from two different booking centres attempt to book four and three seats respectively, at the same instant. It may in such a case very well happen that two seats may be booked in the name of both the parties, creating confusion and trouble at the time of boarding. This will even harm the good name of the company. This type of error may be avoided by allowing access to a particular record by a single user only, at a particular instant of time, i.e., allow resource locking, so that when he is accessing that record, the other requests to that record are queued up, to be served later. Or it may suit the policy of the company to accord different levels of priority to different centres, so that, except while a disk write operation is in progress, the centre with higher priority will push out the others which may be using the system.

The first source of error, as we have already described, cannot be eliminated by any fool proof method. The method for reducing the second source, namely hardware malfunctioning, has been described in great details by Gotlieb and Hume [9]. These methods, though of long time past, are still valid. But since this is not of great importance to our discussion here, we will deal only briefly with it.

A machine's freedome from malfunction is the responsibility of the manufacturer and maintenance engineer, but since the ultimate responsibility for accuracy lie with the programmer, machine reliability is his concern also. Though there is no standard measure of reliability, Mean Time between Failures is assumed to be the standard. During daily maintenance, standard programs, which test different part of the machine are run. The high-speed store, the magnetic drums and disks, the tape units, the input output facilities are all tested with patterns of standard data to provide results which can be easily verified. Another kind of maintenance test of considerable value is marginal checking, first introduced by the group on Whirlwind Computer at MIT. Different sections are run under conditions much more severe than would be expected in a normal run. Faulty components fail and are replaced. This method has become somewhat absolete due to the high costs involved and with the advent of more advanced technologies like TTL circuitry.

Wilkes [10] has dealt with the particular case of maintaining integrity of a data base in the case of hardware failures. Again we are not able to go into the details, but must discuss briefly the main points observed in his treatise.

According to him, there is only one way whereby a high degree of integrity may be achieved in any filing system and this is by keeping a copy of the information to be safeguarded in a separate place, or better still by keeping several copies in several separate places. This, being a very costly affair, cannot really be persued in a large data base system. Next he suggested the process of incremental dumping. Here, any new file created or every new version of a file is only dumped. Even this becomes very costly and so can rarely be used.

For large data bases, that have been in existence for some time, it is usually not possible to completely regenerate the system after error; it must however be possible to repair the data base after an error has been detected. When mag tapes were used for storing the data, it was subdivided into reels, etc., so update or the problem of integrity could be handled in a much simplar manner. But, with the present data bases on direct access media, the problem is complicated by the fact that the need for rapic access with a minimum searching is leading to the use of highly complex data structures, so that there is a problem of maintaining the structural consistency of the data base as well as of maintaining the accuracy of the stored data.

Next he made an attempt to classify and assess the measures that can be taken to monitor the operations being performed on the data base, so that, if information is lost, either the system can recover it automatically, or the user can be assisted to do so. The measures were:

(A) Verification of operations with immediate repetition if in error.

(i) Checks for transient hardware failures (e.g., when writing on mag tape).

(ii) Checking of keyed information

(a) against internal consistency (this implies some redundancy in the keyed information).

(b) against previously recorded information,

(c) for reasonableness.

(B) Redundant (error correcting) coding of information in the data base.

(C) Periodic dumping of the entire data base. This, as pointed out above, is costly and not feasible for large data bases.

(D) The use of journal tapes on which the information is dumped continuously, while the system is in operation.

(i) Transaction journals:

(a) A record of all key strokes made by keyboard operators, editing being either nonexistent or restricted to the deletion from the journal of errors that are immediately corrected.

(b) A condensed summary of the transaction recorded immediately before the update is made.

(ii) Record journals:

(a) before journals, records are dumped before updating.

(b) after journals, records are dumped after updating.

It whould be noted that the effect of a system failure can be to cause an entry in a journal to be incorrectly terminated. The software should be so designed that it is still possible to read the other records on the tape.

For very large data bases, (G) is impractical, so editing of journal tapes is a must. Also, the use of (C) and D(i) makes D(ii) redundant. Greater control should be exercised by the DBA so that consistency can be more easily maintained.

The third source of error, namely, human errors have already been dealt with. Fox programming errors in the DBMS or the underlying operating systems, again one of two procedures are suggested by Gotlieb and Hume [9]. One is to use checking

routines as a monitor during the operation of the program. The other is to use checking routines as a post-mortem or memory dump.

Bayer [11] deals with integrity from the software point of view. He says that the proper use of the system is mainly concerned with quality control in data acquisition and with prevention of accidental or mischeavous misuse, i.e., with the security of the system. Data bases give rise to especia-lly high integrity requirements for at least the following reasons:

[1] Longevity: Even rare errors in the long run will lead to a contamination and deterioration of the quality of a data base. Completely purging erroneous data and all their consequences from a data base is difficult.

[2] Limited repeatability: Even if data or processing errors are discovered, it may be impossible or useless to rectify the situation due to time constraints, unavailability of the correct source data, unavailability of the correct system state preceding the fault, etc.

[3] The need for permanent and immediate availability: This prevents the common practice in other spheres to run a program, debug it, rerun it and so on, to be applicable here.

[4] Multiaccess: Data bases are manipulated by many users having probably quite different quality standards. It is infeasible to completely entrust the quality control to these users and difficult to track the source and the proliferation of errors.

Distinction can be drawn between two forms of integrity, namely semantic and operational integrity.

Semantic Integrity: This is defined as the compliance of the data base contents with constraints derived from our knowledge about the meaning of the data. Semantic integrity might be enforced by allowing on certain data, only a limited set of precisely defined, meaningful operations, by adopting a set of programming and interaction conventions, by dynamically checking the result of the updates, or by proving for each program manipulating the data base, that the semantic intetrity constraints are satisfied.

Operational Integrity: Integrity of a data base must be guaranteed at the beginning and again at the end of transaction, it may be - and generally must be - violated by the single actions. Due to potential interfence between two or more transactions executing in parallel, transactions must lock certain parts for exclusive or shared use.

He then described some algorithms for maintaining operational integrity during parallel operation and resource sharing; involving resource deadlocks and their prevention. We will not discuss the algorithms here.

1-4. GENERAL VALIDATION REQUIREMENTS

The basic process of data validation may be incorporated at three primary sections of the DBMS. These are as follows:

(1) Data validation is highly essential, we may say of critical importance, when we are constructing the data base, i.e., creating the data for the relations which will constitute the data base on the disk pack. Here is the acronym GIGO, mentioned earlier, most appropriate, and so great care must be taken to place only correct (as far as possible), validated data into the data base. The same norms apply when we want to insert new records into an already existing data base.

(2) When we are retrieving a record (or a set of records) for subsequent manipulation, report generation, etc., we need to validate the data.

(3) When updating an already existing record, we need to validate the updated fields of the record, to check whether they cross the acceptable limits and so on, before writing them back onto the disk.

We will now discuss some of the points where data validation is felt to be necessary. First we will enunciate some of the principle points which are valid for any type of data base structure as obtained from Rajaraman's notes [12]. They are:

(1) Primary edit: In this portion of the data validation procedures, the format, i.e., the character type and lengths of the fields are checked. If a record fails to pass this test, it is rejected. Only after this test has been passed is the record presented for the next phase of validation.

(2) Secondary edit: These may fall under several sub-sections:

(a) The values of the data items in some of the domains may lie within certain bounds. These are checked. These fall under limit and plausibility checks.

(b) Some of the fields may be related with one another by some form of relationship. These may be checked.

(c) There may be control totals inserted at the end of the records or a set of records after being calculated off-line. These need to be verified.

Finally, we come to the relational model of data base organisation and the different validation requirements for the data in such a model. Most of these are, of course, also valid for other models. Date [5] suggests the following requirements for validation:

(1) When inserting a new record (or tupле) into the data base, the primary key values in that relation are checked to see that the value of the primary key in this tuple is not equal to an already existing value in the tuple occurrences in the relation, for primary key value in a relation must be unique and so whenever duplication in this domain (or set of domains) is detected, an error signal must be given and operation aborted.

(2) This same check must be applied to all the candidate keys in all the relations, for these again must have unique values.

(3) Again, this check must be made for all domains, or combination of domains (if any, at all), which are supposed to contain unique values.

(4) In third normal form in the relational approach of DBMS, all functional dependencies are eliminated, except for the dependency of the non-key domains on the primary key. But there may still be nonfunctional dependencies, noted as cross references, as for example, the total amount of material issued out of a warehouse should not be greater than the total amount of material put into it. These constraints need to be validated.

The validation requirements enunciated earlier for the general case are also valid here.

Some of the procedures, which we feel would be useful in the above context are stated here. For example, it would be very useful to have presorted relations, since in that case the time for search for a particular data entry will be greatly reduced, as we can use binary search procedures. It would specially be very useful to have an indexed sequential organisation of the data base, with generally the primary keys acting as the index. In such a case, the index table may be brought into main memory, for all the relations being currently accessed, and then a quick binary search of this table alone will help in locating the object of a query and also help in detecting duplicity in primary key, if and

when it occurs, so that an error message could be printed and action taken to avoid it.

For the other candidate keys, we can again keep a directory of these domains for a particular relation and so check for their uniqueness from this directory. For all domains, or combination of domains (again, if any), besides keys, which are supposed to contain unique values, we can perhaps keep a flag where we define these domains, in the field list table, say, to indicate this special property of uniqueness. Then we can check these domains also for their uniqueness. This same procedure may also be adopted in the case of candidate keys other than the primary key.

In the case of nonfunctional dependencies, we can utilise either of two modes of operation, either form a sort of definition table where we name all the different domains which have any kind of nonfunctional dependency between themselves, together with the dependencies; or we can each time specify these dependencies whenever we deal with the specific relations concerned and check for them. In the first method, whenever we come to the specific relation, we look up the particular definition table and finding the various dependencies, validate them by cross checking. In the second method, when we reach the particular position in the tuple, we are given the reference to check and we validate these.

Limit and plausibility checks can be incorporated for specific domains at their particular positions in the typle, i.e., as and when we meet a domain for which limit or plausible value is specified, we validate it for this value without taking into account anything else like its position in the tuple, its relation with other domains, etc. Some of these can be expressed as generalised relationships and can be verified by calls to generalised subroutines, if sufficient number of validations of a particular type are required. An example would be the plausibility checking of the different values of the years in the different data fields in the data base. On the other hand, there are limit and plausibility check requirements for fields which appear only once in each tuple of a particular relation in the data base. These checks can be incorporated in the form of open subroutines in the particular sections involved in the validation of those particular relations.

The next chapter, Chapter 2, deals with the description of the personnel data bases which were selected for the data validation procedures. The various relations, the fields contained in these relations, their dependencies, choice of primary key and the reasons for the choice of the particular structures are described.

Chapter 3 outlines the different validation requirements of the particular domains in the structure chosen. It also describes the various methods found suitable for use in these validations.

Chapter 4 describes the different algorithms developed for satisfying the different validation requirements, as discussed in Chapter 3.

2. NORMALIZATION OF THE PERSONNEL DATA BASE

2-1. INTRODUCTION

For a proper understanding of the process of data validation, which is closely linked with the underlying structure, i.e., with the data base we aim to validate, we would have to study this underlying structure. This chapter attempts to explain the context in which the data validation is proposed to be done, namely the personnel data base which needed to be validated and the basis on which its structure was built up.

As mentioned in the previous chapter our aim was to implement a relational data base management system. For this purpose the data base had to be structured in the form of normalized relations. We will explain in the following section some points about normalization and the advantages of third normal form in which the data base structure was set up. Next we will explain some relevant points including the choice of primary keys, the reasons for the split up of the relations and so on. Finally, we will describe the actual relations themselves.

2-2. NORMALIZATION

The relational approach to data base management is based on the theory of relations which gives it a sound theoretical basis as mentioned earlier. Some valid points about this structure are:

(1) No two tuples (records) are identical in a relation

(2) The ordering of rows (tuples) in a relation is insignificant.

This shows that though a sorted relation is better in terms of query translation, sorting is not an essential part of the structure as such.

(3) The ordering of columns (domains) is insignificant.

This is true assuming reference is made to individual columns by the appropriate domain names never by their relative positions.

The above three points come naturally and are not very significant here.

(4) Every value within a relation - i.e., each domain value in each tuple - is an atomic (nondecomposable) data item (e.g., a number or a character string).

This means that repeating groups are not allowed for a domain value. A relation satisfying property 4 is said to be normalized and the relation thus obtained is stated to be in First Normal Form designated as 1NF.

Now we define full functional dependence a concept important for the later forms. Given a relation R, we say that domain Y of R is functionally dependent on domain X of R if and only if each X-value in Rhas associated with it precisely one Y-value in R. Domain Y is fully functionally dependent on domain X if it is functionally dependent on X and not functionally dependent on any subset of X (it is assumed that X is composite). When we talk now of functional dependence, we will

mean full dependence.

We can now define second normal form or 2NF as:

A normalized relation R is said to be in 2NF if and only if (i) it is in 1NF; and (ii) the non key domains of R, if any, are functionally dependent on the primary key of R.

The 2NF data model suffers from anomalies with respect to storage operations very similar to those encountered with the hierarchical approach, namely those of addition, deletion and updating. These appear due to the fact that some of the non key domains are interdependent. By removing these interdependencies, we arrive at the Third Normal Form, or 3NF defined as:

A normalized relation R is said to be in 3NF if and only if (i) it is in 1NF and (ii) the non-key domains of R, if any, are:

(a) mutually independent
(b) functionally dependent on the primary key of R.

The 3NF representation has the advantage of removing all the three above stated anomalies.

2-3. SOME GENERAL POINTS IN THE PROCESS OF NORMALIZATION

Due to the advantages mentioned above, it was decided to implement the data base in the 3NF representation. We will discuss in this section other factors which led to the present structure.

We will first describe the intial layout of the data from which the data base was proposed to be built up. Figure 2-1 gives a list of all the fields with their serial numbers, names and character type and length. The data, arranged sequentially according to this figure, was originally obtained on a magnetic tape for 2000 officers from the Army HQ EDP Centre. But it was found that the ICL computer used to code the data and load it onto the tape had a very much different character code to our IBM 7044-1401 system through which the conversion of the data and its preparation of output as punched cards was proposed to be done (as the TDC-316, on which this project is proposed for implementation, has no tape units attached in the configuration available to us). So test data for 44 officers were prepared and punched manually and this was done for the different relations in the data base in the final form which we will discuss presently.

The most important criteria for the split up of the data base into the relations was to represent the data in 3NF. Another important criteria was the type of queries expected to be answered by the system. Figure 2-2 gives a list of typical and common queries, their periodicity and their distribution, i.e., from whom it is generated and directed to whom, which determines its importance.

From these queries as well as the layout of the data (in Figure 2-1), it became evident that all the tuples would have to be sequenced in terms of the domain called Personnel

S.No.	Data Items	Picture
1	PARENT ARMS	9(4)
2	PERSONNEL NUMBER	X(7)
3	CHECK DIGIT	A
4	FILLER	X
5	USER-ARM	999
6	TYPE OF COMMISSION	99
7	FILLER	X(6)
8	DATE OF SENIORITY	9(6)
9	FILLER	XX
10	SUBSTANTIVE RANK	X
11	PRESENT RANK	X
12	APPOINTMENT	999
13	DATE OF APPOINTMENT	9(6)
14	SUS-NO.	9(7)
15	COMMAND-CODE	9
16	DATE OF TOS	9(6)
17	MEDICAL CATEGORY	9
18	DATE OF BIRTH	9(6)
19	MARITAL-STATUS	9
20	DATE OF COMMISSION	9(6)
21	TYPE OF CASUALITY	9
22	NAME-F	A(10)
23	NAME-S	A(9)
24	FILLER	X
25	SCHEDULE CASTE-TRIBE	9

Figure 2-1. (cont'd)

26	SS-EC-TO-IC	9
27	FILLER	X(6)
28	STATE-DISTRICT	9(4)
29	CLASS-SUBCLASS	99
30	DATE OF COMMISSION	9(6)
31	DATE OF PRESENT COMMISSION	9(6)
32	SECONDED-TO-CODE	X
33	FILLER	X(4)
34	CAUSE-OF-NE	99
35	DATE-OF-NE	9(6)
36	NCC EXPERIENCE	9
37	FILLER	X(22)
38	DATE OF SUBSTANTIVE RANK	9(6)
39	DATE OF PRESENT RANK	9(6)
40	AUTHORITY OF PRESENT RANK	X(6)
41	PRECOMMISSION STATUS	XX
42	OR-SERVICE	X(5)
43	JCO-SERVICE	X(5)
44	INSTITUTE OF COMMISSION	9
45	TIME SCALE	X
46	FILLER	X(5)
47	RANK	9
48	SHAPE	
	(a) S-VAL	9
	(b) H-VAL	9
	(c) A-VAL	9
	(d) P-VAL	9
	(e) E-VAL	9

Figure 2-1 (cont'd)

49	S-DETAIL	
	(a) S-PERIOD	99
	(b) S-MONTH	X
	(c) S-YEAR	99
50	H-DETAIL	
	(a) H-PERIOD	99
	(b) H-MONTH	X
	(c) H-YEAR	99
51	A-DETAIL	
	(a) A-PERIOD	99
	(b) A-MONTH	X
	(c) A-YEAR	99
52	P-DETAIL	
	(a) P-PERIOD	99
	(b) P-MONTH	X
	(c) P-YEAR	99
53	E-DETAIL	
	(a) E-PERIOD	99
	(b) E-MONTH	X
	(c) E-YEAR	99
54	PST-M	X
55	VALIDATION CODE	X
56	CAUSE-OF-NE-RE-EMP	9(3)
57	DATE OF NE-RE-EMP	9(6)
58	OLD PERSONAL NO.	X(8)

Figure 2-1

Sr. No.	Data Items	Piecture
1	ARM SERVICE AND REGIMENT	9(4)
2	PRESENT ACTING RANK	9
3	PREFIX	XX
4	PERSONAL NO.	9(5)
5	FILLER	X(4)
6	TYPE OF COMMISSION	99
7	DATE OF SENIORITY	9(6)
8	PRESENT SUBSTANTIVE RANK	9
9	MED CAT (OLD FORMAT)	9
10	DATE OF BIRTH	9(6)
11	MARITAL STATUS	9
12	DATE OF COMMN (FIRST)	9(6)
13	NAME	X(20)
14	FILLER	X(4)
15	ACADEMIC QUALIFICATIONS	
	(a) HIGHEST AQ	99
	(b) PHD SUBJECT	99
	(c) MASTER DEGREES(2)	
	(i) SUBJECT	99
	(ii) DIVISION	9
	(iii) FILLER	X
	(d) BACHELOR DEGREE (2)	
	(i) QFN CODE	99
	(ii) SUBJECT (I)	99
	(iii) SUBJECT (II)	99
	(iv) SUBJECT (III)	99
	(v) Percent MARKS	9
	(vi) DIVISION	9
	(vii) SUBJECT (IV)	99

Figure 2-1 (cont'd)

16	MEMBERSHIP OF INSTITUTIONS (6)	X(3)
17	PROMOTION EXAMS	
	(a) PART A	X
	(b) PART A TECH (Now FILLER)	X
	(c) PART B	X
	(d) PART C	X
	(e) PART C TECH	X
	(f) PART D	X
18	PROFESSIONAL/TECHNICAL QUALIFICATIONS	(5)
	(a) QFN CODE	999
	(b) INSTITUTION	999
	(c) YEAR	99
	(d) SUBJECT(I)	99
	(e) SUBJECT (II)	99
	(f) MARKS	9
	(g) DIVISION	9
19	LANGUAGES	
	(a) FOREIGN (4)	A
	(i) LANGUAGE CODE	AA
	(ii) STANDARD	9
	(iii) PROFICIENCY	9
	(iv) FILLER	X
20	MOTHER TONGUE	AA
21	ARMY COURSES (15)	
	(a) NAME	X(4)
	(b) GRADING	XXX
	(c) YEAR	99
	(d) DURATION	999
	(e) FILLER	XX
22	HONOURS AND AWARD (6)	A(4)
23	QFN PAY	9
24	MED CATEGORY	
	(i) S (ii) H (iii) A (iv) P (v) E	9(5)

Figure 2-1 (cont'd)

25	DATE OF NE (IN NE-FILE ONLY)	9(6)
26	DATE OF PRESENT RANK	9(6)
27	DATE OF SUB RANK	9(6)
28	FILLER	X(49)

Figure 2-1

Number. This would be true for all relations in the data base. In many of these relations, this domain would be the primary key as well, satisfying the uniqueness property of primary keys. But in some of the relations it was also evident that the Personnel Number alone could not serve as the primary key because the above mentioned constraint cannot be satisfied in these cases. Let us take an example of such a relation. Let us take the case of an army officer possessing several civil qualifications like. B.A., M.A., L.LB., etc. For the sake of normalization, we would have to put each qualification in a separate tuple, yet each will be associated and primarily depend upon the Personnel Number of that officer only. So we would be repeating the same value of this domain in several successive tuples and hence by the property of uniqueness, it cannot be the primary key in such relations. It was found that a combination of this domain with some other domain, like the qualification code in the above case, could serve as the primary key in such relations, satisfying the criterion of uniqueness. In other relations, where only unique values could exist in other domains for a particular personnel number, like an officer's name, date of birth, date of first commissioning, etc., it was found possible to satisfy the constraint by this domain alone and this caused its adoption as the primary key in those relations.

Thus we find that the choice of the primary key split the data base into two groups of relations, one with a single domain

Sl. No.	Title of the Reports (Queries)	Periodicity	Distribution
1	Actual strength of officers by parent arms/services and type of commission	Monthly	Org 2 MS Coord ASO Man
2	Increase/decrease in the strength of officers by arms/service, type of commission and causes	Monthly	Org 2 AG Budget ASO Man
3	Actual strength of officers by user/parent arms/services	Quarterly	Org 2 ASO Man
4	Actual strength of officers by Arms/Services and Ranks	Quarterly	Org 2 MS coord Miscellaneous
5	Actual strangth of officers by units	Yearly	ASO Man
6	Actual strength of officers holding ERE unspecified appointment by arms/services and serving in regular army and other than regular army units	Monthly	ASO Man
7	Appendix (split by ranks) to actual strength of officers holding ERE unspecified appointment by arms/services and serving in regular army and other than regular army units	Half-yearly	ASO Man
8	Actual strength of regular army officers serving in TA by parent arms, commands and units	Monthly	ASO Man
9	Appendix (split by Ranks) to actual strength of regular army officers serving in TA by parent arms, commands and units	Half-yearly	ASO man
10	Actual strength of officers holding SL commission by user and parent arms and by regtt., ERE specified and ERE unspecified appointment	Monthly	ASO Man

Figure 2-2 (cont'd)

11.	Appendix (split by ranks less QMS and ROs) to actual strength of officers holding SL commission by user and parent arms and by Regtt. ERE specified and ERE unspecified appointments	Half-yearly	ASO Man
12.	Details of changes in the strength of officers by causes giving No., Name, Rank and effective date of change	Quarterly	PSS
13.	Actual strength of regular army officers with HQ and units of DGBR by ranks and commands	Quarterly	ASO Man
14.	Actual strength of ARO's and BRO's by arms/services and classes	Yearly	Org 2 AG Budget ASO Man
15.	Nominal roll of PC officers due for subs promotion	Yearly	MS3
16.	Nominal roll of non-regular officers due for promotion	Yearly	MS3
17.	Nominal roll of officers (SL) due for promotion	Yearly	MS3
18.	Nominal roll of officers excluding re-employed, RRO's and TCOs service in civil units and Military Missions abroad	Quarterly	PS5 ASO Man
19.	Actual strength of officers by year of birth and year of seniority	Qrtly.	ASO Man
20.	Actual strength of officers serving in army HQ, Inter services Organisations and Units, Ranks and Marital Status	Qrtly.	ASO Man
21.	Nominal roll of officers placed in low medical category	Half Yearly	DGAFMS (DG-3) DMS 5 ASO Man
22.	Actual strength of non-medical officers placed in low medical category by arms/Services, rank and medical category	Half yearly	Org 2 DMS3 ASO Man
23.	Actual strength of Engrs, Signs., and EME officers by arms/services, type of commission and educational qualifications	Half yearly	Org 2 ASO Man

Figure 2-2 (cont'd)

24.	Nominal roll of AMC, ADC officers by units, ranks and personal numbers	Half yearly	DMS1 ASO man
25.	Actual strength of officers by arms/services serving in army HQ, Military Missions, commands and other miscellaneous units of the regular army	Half yearly	ASO Man
26.	Actual strength of officers belonging to SC/ST by arms/services for regular and other than regular army.	Yearly	ASO Man
27.	Fresh-in-takes of officers belonging to SC/ST by arms/services, types of commission and state of domicile for regular and other than regular army during the year 1 Jan. to 31 Dec.	Yearly	ASO Man
28.	Nominal roll of regular army officers in personnel number order giving personal number, rank, name and present unit	Half yearly	APS PS5 PS8 Org Coord.
29.	Nominal roll of officers possessing legal qualifications, by formations	Yearly	JAG Dept. HQ comds. HQ Corps. Div. and Area HQ
30.	Actual strength of officers on X-list by arms/services, ranks and causes	Half yearly	ASO Man
31.	Nominal roll of regular army officers in alphabetical order giving personal number, rank, name and present unit	Half yearly	APS Miscellaneous ASO Man
32.	Nominal roll of officers serving with specified units and establishment	Half yearly	AG Budget
33.	Actual strength of officers by subs rank, type of commission, year of birth and seniority combined and for each separately	Yearly	MS Coord Org. 2 ASO Man
34.	Actual strength of substantive majors and below by ranks, year of birth and year of commission combined and separately for each arm.	Yearly	MS Coord Org. 2 ASO Man

Figure 2-2 (cont'd)

35.	Actual strength by arms/ services and classes	Yearly	MS coord Org 2 ASO Man
36.	Actual strength by arms/ services and marital status	Yearly	ASO Man
37.	Actual strength of officers by units and parent arms/ services in HQ form, Misc., Units and ARA units (for ARA units by parent arm, ranks in seprrate ppx)	Half-yearly	ASO Man
38.	Actual strength of regular army officers in ARA units and X-list (separately) for regular and outside regular army units by parent arms	Monthly	ASO Man
39.	Actual strength of officers by present ranks and age groups	Quarterly	ASO Man
40.	Nominal roll of officers due for long service medal (a) 9-years (b) 20-years	Yearly	Org-3 Org-9 Med Dt. (for AMC/ ADC offrs)
41.	Actual strength of officers by user arms/and present ranks	Quarterly	ASO Man
42.	Actual strength of offrs. by marital status for medical and non-medical separately	Quarterly	ASO Man
43.	All officers courses	Yearly	MS Branch
44.	Strength of Engrs, Sigs. and ERE Offrs by qualifications	Yearly	MS Branch Org. 2
45.	PC Officers with 20 years service who have not passed part D examination	Yearly	MS8

Figure 2-2

as the primary key while in the other a combination of domains acted as the primary key. The consideration of the typical queries and their frequencies (in Figure 2-2) formed another basis for the division of the data. Final basis was the interdependence between non-key domains which caused further subdivision. The data base was split into twelve important relations based on the above criterion, covering almost all the domains specified in the list of Figure 2-1. The few domains which were left out were ignored either because they do not form any part of the queries and were determined to be relatively unimportant or else their meaning was unexplained and possibly highly confidential.

2-4. THE FINAL STRUCTURE

The greatest number of queries, with also the highest frequency rate, was observed to be according to the domain called Corps. So relation called ARM was formed having this domain as the major part of the primary key, though the sequencing of the tuples was still done according to the domain of Personnel Number which formed the minor part of the key. This relation is unique in this respect as in all the other relations, Personnel Numbers forms the primary key alone or is a major part of the primary key for the purpose of search for data retrieval. Rank and Time Scale were the other two domains which were found to be related by queries to this relation and were thus added. This relation is shown as relation 1 in Figure 2-3.

Relation 1: ARM

Personnel No.	Arms/Service or Corps	Rank code	Time Scale
(7 bytes alphanumeric)	(2 bytes, numeric)	(1 byte numeric)	(1 byte, alpha)
IC-12343	07	3	Y
IC-21001	02	5	N

Relation 2: NAME

Perso-nnel No.	Name-S	Name-F	Date of birth	Marital status		SC/ST?	Reli-gion
7 bytes, alpha-numeric	Alpha-betic, 16 bytes	Alpha-betic, 24 bytes	Numeric 6 bytes	Alpha 16 bytes		Alpha 1 byte	Alpha 1 byte
IC10000	RAINA	THAR NARA-SIMHA	25.11.22	Y		N	H
IC10060	PATHAK	GOPAL SWARUP	270324	Y	7	N	H

Perso-nnel No.	Mother Tongue code	Home state code	If will made	If quali-fied	If honoured	If Secon-ded to R and D
7 bytes, alpha-numeric	Numeric 2 bytes	Numeric 2 bytes	Each of length 1 byte and type alphabetic			
IC10000	02	01	Y	Y	Y	N
IC10060	02	01	Y	N	Y	N

Figure 2-3 (cont'd)

Relation 2: NAME

Personnel No.	If seconded DGI	Legal qualification?	Foregin language known?	If member of Inst.	No. of children	No. of dependents
	Each of length 1 byte and type alphabetic				Num. 1 byte	Num. 1 byte
IC10000	N	N	Y	Y	3	7
IC10060	N	N	Y	Y	2	4

Relation 3: COMM

Personnel No.	Date of 1st comm.	Date of change to PC	Date of seniority	Precom. status code	Other rank service	JCO service	Inst. of comm.
7 bytes alpha-num.	Num. 6 bytes	Num. 6 bytes	Num. 6 bytes	Num. 1 byte	Num. 4 bytes	Num. 4 bytes	Alpha 3 bytes
IC10050	510614	510614	490614	1	2301	0310	IMA
IC12002	520529	570420	520529	8	2200	0603	OTS

Personnel No.	If Re-emp.	NCC experience
	Alpha 1 byte	Num. 1 byte
IC10050	Y	200
IC12002	Y	100

Relation 4: QUAL

Personnel No.	Civil Degree code	Main subject code	Div. obtained	Percent marks	Year passed out
7 bytes alpha-num.	Num. 2 bytes	Num. 2 bytes	Num. 1 byte	Numeric 2 bytes	Numeric 2 bytes
IC31233	07	27	2	56	58
SS13987	08	21	1	79	70

Relation 5: LANGI

Personnel No.	Code of Indian Language	Degree of Proficiency	If Exam. Passed
Alpha-num. 7 bytes	Num. 2 bytes	Num. 1 byte	Alpha 1 byte
IC10100	12	4	Y
IC15400	03	5	Y

Relation 6: LANGF

Personnel No.	Code of Foreign Language	Level of Exam. Passed	Year Passed
Alpha-num. 7 bytes	Num. 2 bytes	Num. 1 byte	Num. 2 bytes
IC10061	03	3	57
IC27001	04	1	69

Figure 2-3 (cont'd)

Relation 7: PQUAL

Personnel No.	Prof. course code	Grading code	Year Passed
Alpha-num. 7 bytes	Num. 2 bytes	Alpha 1 byte	Numeric 2 bytes
IC10052	11	A	60
IC12140	08	B	54

Relation 8: MED

Personnel No.	S				H			
	Period	Month	Year	Value	Period	Month	Year	Value
Alpha-num.	All numeric				All numeric			
7 bytes	2 bytes	2bytes	2b	1byte	2byte	2byte	2byte	1byte
IC19106	15	02	76	1	14	03	76	1
SS13987	15	02	76	2	15	02	76	1

A				F			
Period	Month	Year	Value	Period	Month	Year	Value
All numeric				All numeric			
2bytes	2bytes	2byte	1byte	2bytes	2bytes	2b	1 byte
15	02	76	1	15	02	76	1
15	02	76	1	15	02	76	1

E			
Period	Month	Year	Value
All numeric			
2bytes	2bytes	2bytes	1byte
14	03	76	1
15	02	76	2

Figure 2-3 (cont'd)

Relation 9: UNIT

Personnel No.	SUS No.	Command Region	Date of TOS	Appt Code	C From Exam.	D From Exam.	Substantive rank code
Alpha-num. 7 bytes	Num. 7 bytes	Alpha 1 byte	Num. 6 bytes	Num. 1 byte	Alpha 1 byte	Alpha 1 byte	Num. 1 byte
IC12408	0456654	S	760809	2	Y	N	3
SS16654	0632123	E	760910	3	N	N	9

Personnel No.	Date of subs. rank	Present rank code	Date of Pr. rank	ERE Appt?	TA Appt?
Alpha-num. 7 bytes	Num. 6 bytes	Num. 1 byte	Num. 6 bytes	Alpha 1 byte	Alpha 1 byte
IC12408	760712	3	760712	Y	N
SS16654	760601	9	760601	N	N

Relation 10: MINST

Personnel No.	Name of Instt. Code	Type of Membership Code
Alpha-num. 7 bytes	Num. 2 bytes	Num. 1 byte
IC24343	13	3
SS13877	21	2

Relation 11: REMP

Personnel No.	New Pers. No.	Date of Retirement	Type of Release code	Caurse of Release	Date of reemployment
Alpha-num 7 bytes	Alpha-num. 7 bytes	Num. 6 bytes	Num. 1 byte	Alpha 26 bytes	Num. 6 bytes
IC10000	IC10000	760902	1	Age limit reached	760903

Figure 2-3 (cont'd)

Relation 12: DECOR

Personnel No.	Award Name Code	Year awarded	Comnand Region
Alpha-num. 7 bytes	Num. 2 bytes	Num. 2 bytes	Alpha 1 byte
IC10000	12	63	N
SS12787	06	70	C

Figure 2-3 (cont'd)

The next relation was called NAME and it contains all the personal details for a particular officer. Originally, it had been proposed that this relation should be broken up into two, named NAME and PERS, to cater to queries of two different types and frequencies. But since then it was found that domains like the officer's address, his NOK's name, his address, etc., which were proposed to be put into the second of the two relations, namely PERS, and would have made the tuples of that relation quite long and so undesirable for retirieval every time, were not present in the supplied data domains. So the size of this relation was drastically reduced. There were also many domains common between the two relations so that it was now found viable as well as desirable for the sake of minimum data redundancy to combine the two relations and call it NAME. But we strongly recommend that the domains which were sought to be put into this relation and could not be because they were missing from the original supplied data should be included in the Army Officers Records. The primary key in NAME is Personnel Number. It contains all the personal details of an officer which could be generally asked in a query and is the largest relation (in terms of tuple length) in our data base. It also contains a large number of pointer domains which indicate the presence of tuples in other relations.

Next came the relation COMM which contains the commissioning details of each officer. Here again, since one and only one tuple exists for each officer, Personnel Number was suitable

as the primary key. The other related domains, according to queries, were the First-Date-of-Commissioning, the Date-of-change-to-PC, the Date-of-seniority, Precommission status code, OR-service, JCO-service, Institute of commissioning, whether Re-employed and NCC experience. They were assembled in the above given order in this relation. This relation is relation 3 in Figure 2-3.

Next comes the relation QUAL, shown as relation 4 in Figure 2-3. This relation contains the civil qualifications of the army officers. It was found, as discussed earlier, that a single officer may have several civil qualifications. So, in the case of this relation, the domain Personnel Number alone could not act as the primary key. But a combination of this domain, with the domain for the civil degree code was found to contain unique values and since the other related domains were found to be functionally dependent on the combination it was found suitable to adopt this as the primary key for this relation. The other related domains were subject code, division, percentage marks and the year in which passed out. Presnece of a tuple in this relation for a particular personnel number is indicated by a pointer domain in the relation NAME called IF-Qual.

The next relation in Figure 2-3 is relation 5, named LANGI. This relation contains the details of the Indian Languages known by the officers. Here again, for reasons similar to above, a combination of the domains of Personnel

3. VALIDATION REQUIREMENTS AND PROCEDURES

3-1. INTRODUCTION

We have already discussed, in the previous chapter, the various relations that were decided upon as the constituents of our data base, together with the domain combinations in them. We also discussed the reasons whereof this particular structure was chosen and decided upon the domain or domain combination acceptable as the primary key for each relation. This chapter attempts to describe the various validation requirements of each relation, first in general terms and then on a relation by relation basis, and the methods used in achieving these. As stated by Date [5], it is not possible to check each and every item of data in a data base and maintain an absolute integrity. What has been attempted is to keep the data as error free as possible when storing it in the data base (in our case we had to assume that the data supplied was error free, because we have no means of checking back), to detect any errors, if and when they occur, when we retrieve this data for subsequent manipulation or report generation through simple but effective methods which would not require appreciable computer time or much extra storage space, and in the case of a few domains, where it was thought desirable to preserve a more rigid control on the integrity of the values of the data elements, simple coding was attempted in a form that

facilitated error detection and in one single case also provided for error correction.

Data need to be validated in four stages of a data base system implementation. The four stages are depicted by the block diagrams in Figure 3-1. As we observe here, the requirements for validation in the first and third stages, namely during initial data base building and during data storage after manipulation, are similar while those in the second and 4th stages, namely during data retrieval and for report generation are two complementary activities. So the validation would be treated as essentially parts of the first and second stages and differences, whenever they occur, between these two and the other two will be indicated.

3-2. INPUT DATA VALIDATION

3-2.1 Primary Edit: We will start with the discussion of the first stage. As we have noted earlier, in the first chapter, description of data validation must begin with the primary edit facilities provided by the system. For our purpose we had chosen four allowable format types, namely decimal integer, binary integer, alphabetic and alphanumeric. These were allotted the code values of 1,2,3 and 4 respectively for input to the computer but inside the computer these were transformed to 0,1,2 and 3 respectively to conform to the format specifications being used by the person utilising the data for the data base BUILD function. The format specifications for each relation was to be given in the form of

Number and code of language was chosen as the primary key. The other related domains were proficiency and whether any examinations were passed in the subject.

Relation 6 in Figure 2-3 is the relation LANGF, containing details of the foreign languages known by the army officers. Here again, to satisfy the property of uniqueness, a combination of the domains of Personnel Number and the code of Language was selected as the primary key. The other related domains were the Highest-Examination-Passed and the Year-Passed.

Then came the relation PQUAL, shown as relation 7 in Figure 2-3. This relation contains a list of all the courses attended by the officers after commissioning in the army. Here again, the same difficulty as related to primary keys arose, as in the previous three relations, because each officer may and does attend a large number of courses and so in the normalized form Personnel Number is repeated as many times as the number of courses attended by the officer. A similar combination of this domain and the professional course code, satisfying the property of uniqueness, was chosen as the primary key. The other related domains were the Grading Code and the Year-Passed-Out.

Relation 8 in Figure 2-3 is the relation MED. This relation contains the details of the medical biodata of the officers, i.e., their medical fitness records. This relation can have only single tuples for each person and so Personnel Number could act here as the primary key. The medical fitness tuple for any

officer is split into five blocks, each block containing the four domains of Period, Month, Year and Value, in that order respectively. The blocks are named S,H,A,P and E, in that order respectively, giving rise to the acronym SHAPE.

Next to come in Figure 2-3, is relation 9, called UNIT. This relation contains the details of each officer as related to the unit to which he is posted. Here again, each officer being related to one and only one tuple, personnel number was found suitable to act as the primary key. It contains besides the domains named SUS Number, Command Region, Date-of-TOS, Appointment-code, C-Promotions Examinations, D-Promotion-Exams., Substantive Rank, Date-of-subs-Rank, Present Rank, Date-of-P-Rank, If-ERE-Appointment and If-TA-Appointment.

Relation 10 which comes next is called MINST. This relation contains the details of officers who are members of different Institutions and Socieities, professional and otherwise. Since the same officer could be member of several Institutions at the same time, we were forced again to seek a combination of the domains of personnel number and name-of-Institute-code to act as the primary key. The only other domain found linked to this relation was the Type-of-Membership code and was added. This relation is again related to the relation NAME by means of a pointer domain in the latter relation.

Relation 11 in Figure 2-3 is called REMP. As its name suggests, it contains the details of all officers who are reemployed after retirement by the army. The primary key in

this case is personnel number alone. The other domains are new personnel number, date-of-retirement, type-of-release, cause-of-release and the date-of-reemployment.

The last of these relations of Figure 2-3 is relation 12 called DECOR. This relation contains the details of all officers decorated for showing efficiency in organization or bravery in the field of battle. Here again, a single officer could have obtained a number of decorations and so personnel number alone fails to be the primary key. A combination of personnel number and the award name code satisfies the uniqueness property as well as the requirement of functional dependence of other domains and was chosen to play the part of primary key. The other related domains are those of year received and command region in which the officer was serving at the time of docoration.

Figure 2-4 gives a list of all the relations in a form of relation table. The contents of this table are the serial number of the relations, their names and all the identifications of the various domains contained in them. Figure 2-5 contains a list of all the domains in the data base in the form of a domain table. The contents of this table are the domain identifiers in serial order, their names, their types and their lengths. Figure 2-6 gives the list of other tables used in supplantting some of the domains in the data base, i.e., to define the various values and to give the meanings of the various codes used in the different domains of the data base. Finally, Figure 2-7 gives details of these tables.

Relation Table:

Code	Relation Name	Fields contained
R1	ARM	F1,F2,F4,F6
R2	NAME	F1 to F5,F7toF12, F22toF28.3
R3	COMM	F1,F2,F4,F13 to F21
R4	QUAL	F1,F29 to F33
R5	LANGI	F1,F34,F35,F36
R6	LANGF	F1,F37,F38,F39
R7	PQUAL	F1,F40,F41,F42
R8	MED	F1,F43, to F62
R9	UNIT	F1, F63 to F74
R10	MINST	F1,F82, F83
R11	REMP	F1,F75 to F79
R12	DECOR	F1,F80, F81, F64

Figure 2-4

Sr No	Name	Type	Length
F1	Personnel No.	AN	7 bytes (before) 8 bytes (after)
F2	Rank	N	1 byte
F3	Name-S	A	16 bytes
F4	Arms/Service	N	1 byte
F5	Name-F	A	24 bytes
F6	Time Scale (Y/N)	A/N	1 byte
F7	Date-of-Birth	N	6 bytes
F8	Marital status (Y/N)	A/N	1 byte
F9	If-Qualified (Y/N)	A/N	1 byte
F10	If-on-Honours (Y/N)	A/N	1 byte
F11	If-seconded-to-R and 1)(Y/N)	A/N	1 byte
F12	If-seconded-to-DGI (Y/N)	A/N	1 byte
F13	Date-of-1st-commissioning	N	6 bytes
F14	Date-of-change-to-PC	N	6 bytes
F15	Date-of-seniority	N	6 bytes
F16	Precomm-status-code	N	1 byte
F17	Other-rank-service	N	4 bytes
F18	JCO-Service	N	4 bytes
F19	Institute-of-commissioning	A	3 bytes
F20	If-RE-EMPLOYED (Y/N)	A/N	1 byte
F21	NCC-Experience	N	3 bytes
F22	If-SC/ST (Y/N)	A/N	1 byte
F23	Mother tongue code	N	2 bytes
F24	Home state code	N	2 bytes
F25	Religion	A	1 byte
F26	If-Will-Is-Made (Y/N)	A/N	1 byte
F27	If-Member-of-Any-Institute(Y/N)	A/N	1 byte
F28	If-Having-Legal-Qualification (Y/N)	A/N	1 byte
F29	Civil-Degree-Code	N	2 bytes
F30	Subject-code	N	2 bytes
F31	Division	N	1 byte
F32	Percent Marks	N	2 bytes
F33	Year-passed-out	N	2 bytes
F34	Code-of-Indian-Language	N	2 bytes
F35	Proficiency code	N	1 byte
F36	If-Exam-Passed (Y/N)	A/N	1 byte
F37	Code-of-Foreign-Language	N	2 bytes
F38	Level-of-Exam-Passed	N	1 byte
F39	Year-passed	N	2 bytes
F40	Prof-course-code	N	2 bytes
F41	Grading code	A	1 byte
F42	Year-pa ssed	N	2 bytes
F43	S-Period	N	2 bytes
F44	S-Month	N	2 bytes
F45	S-Year	N	2 bytes
F46	S-Value	N	2 bytes
F47	H-Period	N	2 bytes
F48	H-Month	N	2 bytes
F49	H-Year	N	2 bytes
F50	H-Value	N	1 byte

Figure 2-5 (cont'd)

F51	A-Period	N	2 bytes
F52	A-Month	N	2 bytes
F53	A-year	N	2 bytes
F54	A-value	N	1 byte
F55	P-Period	N	2 bytes
F56	P-Month	N	2 bytes
F57	P-Year	N	2 bytes
F58	P-value	N	1 byte
F59	E-Period	N	2 bytes
F60	E-Month	N	2 bytes
F61	E-Year	N	2 bytes
F62	E-Value	N	1 byte
F63	SUS-No	N	6 bytes
F64	Command-Region	A	1 byte
F65	Date of TOS	N	6 bytes
F66	Appointment code	N	1 byte
F67	C-Promotion Exam. (Y/N)	A/N	1 byte
F68	D-Promotion Exam. (Y/N)	A/N	1 byte
F69	Substantive rank	N	1 byte
F70	Date of subs rank	N	6 bytes
F71	Present rank	N	1 byte
F72	Date of present rank	N	6 bytes
F73	ERE appointment (Y/N)	A/N	1 byte
F74	TA appointment (Y/N)	A/N	1 byte
F75	New Personnel No.	A/N	7 bytes (before) 8 bytes (after)
F76	Date of Retirement	N	6 bytes
F77	Type of Release	N	1 byte
F78	Cause of Release	A	
F79	Date of Reemployment	N	6 bytes
F80	Award name code	N	2 bytes
F81	Year awarded	N	2 bytes
F82	Name of Institute code	N	2 bytes
F83	Type of membership code	N	1 byte
F28.1	Knowing foreign language (Y/N)	A	1 byte
F28.2	No. of children	N	1 byte
F28.3	No. of dependents	N	1 byte

Figure 2.5

Table of Tables

Code	Name
T1	Rank Code
T2	Mother tongue code
T3	Home State code
T4	Precomm status code
T5	Civil degree code
T6	Main subject code
T7	Foreign-language code
T8	Professional courses code
T9	Tupe of release code
T10	Award name code
T11	Name of Institute code
T12	Type of Membership code

Figure 2-6

Table T1: Rank Table

Rank Code	Designation
0	General
1	Lt. General
2	Maj. General
3	Brigadier
4	Colonel
5	Lt. Colonel
6	Major
7	Captain
8	Lieutenant
9	Second Lt.

Table T2

Code	Name
01	English
02	Hindi
03	Bengali
04	Tamil
05	Telegu
06	Kannar
07	Marathi
08	Malayalam
09	Gujarati
10	Orriyya
11	Assamese
12	Gurumukhi
13	Kashmiri
14	Urdu
15	Farsi
16	Sindhi
17	Sanskrit

Table T12

Code	Designation
1	Graduate
2	Fellow
3	Associate
4	Student
5	Full
6	Honorary

Table T3

Code	State Name
01	U.P.
02	M.P.
03	Bihar
04	Rajasthan
05	West Bengal
06	Maharashtra
07	Gujarat
08	Orissa
09	Assam
10	Punjab
11	Haryana
12	Tamilnadu
13	Karanataka
14	Kerala
15	
16	Andhra
17	Himachal Pradesh
18	Tripurra
19	J and K
20	Arunachal
21	Meghalaya
22	Nagaland
	Union Territories and Misc.

Table T5

Code	Degree Name
01	B.A.
02	M.A.
03	B.Sc.
04	M.Sc.
05	B.Com.
06	M.Com.
07	LL.B.
08	B.E.
09	M.E.
10	B.Tech.
11	M.Tech.
12	Ph.D.
13	Miscellaneous

Table T7:

Code	Name of Language
01	Russian
02	German
03	French
04	Chinese
05	Japanese
06	Spanish
07	Portuguese
08	Arabic
09	Persian
10	Greek
11	Italian
12	Burmese
13	Miscellaneous

Table T8

Code	Name of Course
01	P.Sc.
02	JC
03	EME Tels
04	Commando
05	Sr. Cammirssioning
06	Young Officers
07	JAWS
08	Paratroopers
09	LGSC
10	PTSC
11	Company commander
12	Catering
13	Intelligence
14	PT
15	Weapon training
16	Snow warfare
17	Montainceering
18	Short equipment
19	Senior Ammunication Tech.Officers
20	NDC
21	Defence Mgmt.
22	Work Study
23	Tank Technology
24	Miscelleneous

Table T4

Code	Designation
1	Student
2	Clerk
3	Salesman
4	Apprentice
5	Storekeeper
6	Scientist
7	Engineer
8	Miscellaneous
9	Teacher

Table T9:

Code	Type of Release
1	Normal Release
2	Premature Retirement
3	Cashiering
4	Dismissal
5	Medical

Code	Name of Award
01	Ashok Chakra III
02	Ashok Chakra II
03	Ashok Chakra I
04	Shawraya Chakra
05	VSM
06	AVSM
07	PVSM
08	Mentioned in Despatch
09	SM
10	VC
11	MVC
13	PVC

Table 2-7 (cont'd)

Table T6

Code	Name of the Main Subject
01	Economics
02	English
03	Philosophy
04	Political Science
05	Varnacular
06	Physics
07	Chemistry
08	Mathematics
09	Zoology
10	Botony
11	Statistics
12	Geology
13	Accounts
14	Hydraulics
15	Environmental
16	Civil
17	Aeronautics
18	Chemical
19	Mechanica l
20	Electrical
21	Electronics
22	Telecommunication
23	Architectural
24	Naval architectura l
25	Mining
26	Metallurgy
27	Miscellaneous
28	Computer Science

Table T11

Code	Name of Instt (or society
01	CSI
02	IEEE
03	Instt. of Engg. (India)
04	ASC
05	IGS
06	FIP
07	I. Math. S.
08	Aero Soc. of India
09	All India Mgmt. Assoc.
10	American A. for Ad.ofSc.
11	Am.A.for Petroleum Geo.
12	Am.Geographical Soc.
13	AIChE
14	AIEE
15	A.Math.Soc.
16	Am.Inst. of Chemists
17	A.S. of Civil Engg.
18	ASME
19	ASNE
20	Asiatic S. of Bengal
21	As. of Applied Physicist
22	ACM
23	Calcutta Hist.Soc.
24	Calcutta Math. Soc.
25	Am. Inst. of Architects
26	Red Cross
27	IFF
28	IHF
29	Indian S. of Civil Engg.
30	IIL
31	IIC and WA
32	NIS
33	Miscellaneous

Figure 2-7

BLOCK DIAGRAM FOR THE INITIAL PHASE OF DATA BASE BUILDING OR ADDING RECORDS

DATA READ

SELECT RELATION

PRIMARY EDIT

CHECK PLAUSIBILITY, LIMITS, etc.

CHECK INTERFIELD RELATIONSHIPS

CONVERT Y/N FIELDS

COMPUTE NECESSARY CHECKSUMS

PLACE IN APPROPRIATE BUFFER

PLACE ON DISK

BLOCK DIAGRAM FOR DATA OUTPUT

TRANSFER TO BUFFER

VALIDATE CHECKSUMS

DECONVERT Y/N's

REMOVE CHECKSUMS, etc.

PRINT

Figure 3-1

BLOCK DIAGRAM FOR DATA RETRIEVAL

QUERY
↓
SELECT RELATION
↓
READ FROM DISK
↓
VALIDATE PRIMARY KEY
↓
CHECK SUMS VALIDATE (If any)
↓
TRANSFER TO BUFFER

BLOCK DIAGRAM FOR DATA STORAGE AFTER MANIPULATIONS

TRANSFER TO BUFFER
↓
PRIMARY EDIT
↓
CHECK FOR DUPLICATION OF PRIMARY KEY
↓
CHECK INTER-FIELD RELATIONSHIPS
↓
CHECK FOR PLAUSIBILITY, LIMITS, etc.
↓
CHECK Y/N FIELDS AND IF NECESSARY CONVERT
↓
RECOMPUTE CHECKSUMS
↓
WRITE ON DISK

Figure 3-1

consecutive 3 digit decimal numbers for each domain, the first digit giving the domain type and the other two digits giving the length of the domain as the number of characters or bytes. Inside the computer, these format specifications were stored in one word, with the first 4 bits of the word corresponding to the domain type and the rest twelve bits used for storing the length.

Whenever data was being stored, either during the initial phase of data base building or later when data is being inserted into an already existing data base, the first card was used to specify the format of the relation being currently stored. These formats are suitably converted and stored by a routine called PEFORM. Next comes a card that gives the number of records being stored in that relation and this was checked and stored by a routine called NOFREC. Now, when the actual tuples of the relation are read in, they are checked for format validity (according to format description stored by PEFORM) by a routine called PECHECK. Any error during this primary edit procedure gives rise to an error message and causes the offending tuple to be printed out as well.

3-2.2 Primary Key Validation: At the next step we start our secondary edit procedures. The most important among these and one that concerns us in all the relations is the validation of primary keys to prevent their duplication. This was achieved in our case by a routine called PKCHEK. This routine compares the primary key of a tuple with the primary keys of the previousl

stored tuples, which are separately stored by this routine itself, and if the key is found to be unique, it adds it to its storage of primary keys and passes control back to the main routine for further processing. In case of duplication occurring it prints an error message followed by the offending tuple and waits for restart so that the card containing the tuple can be replaced. In our case, the relations were structured in such a fashion as to prevent the occurrence of candidate keys other than the primary key, so there was no need of any other type of key validation.

3-2.3 Validation of Some General Domains: Next we come to a discussion of some particular domains or domain types. First, and foremost in our discussion must come the domain named Personnel Number. This domain is very important because as we found in the previous chapter, it forms a part of every relation in our data base and is the primary key in many of these relations and is a vital part of the primary key in all the other relations. This domain has as input a seven-byte alphanumeric string, the first two bytes being characters from the alphabet while the remaining five bytes are decimal digits. The two-character first section is chosen from a small set of particular combinations of letters, and so can be easily validated. This is initially done by a routine called PNCV. Next the five-decimal-digit section was coded in a simple manner by another routine named PNDC. This converted it to a six-digit figure instead of its original five bytes. This was done with two specific aims in mind. One was to diffuse

or scramble the value, so that without the help of the proper coding function, the data value in this domain cannot be matched and so the tuple cannot be retrieved. This would provide a simple method of preventing illegal access to the data as all accesses to the data base by primary key matching would be affected as this domain is the primary key or a vital part of it for all the relations in the data base, as discussed earlier. The second aim is to provide some measure of error detection capabilities, in accordance with the importance of this domain.

The domain or domain type considered the most important after this was the type of domain which contained the simple binary answer of Y or N (standing for Yes and No respectively). This domain type was considered important for two reasons. First is its high rate of occurrence in our data base. A total of 17 domains out of a total of 83 domains, giving a proportion of 1 in 5 roughly, were found to be of this type. The other factor contributing to the importance of this type of domain is the fact that in many cases it acted as a pointer to the presence or absence of tuples in other relations in the data base. We will illustrate this point with an example. In the relation called NAME, there is a domain called If-Qualified. This is a domain of the above type, indicating whether that particular person has any civil qualification or not. The value Y indicates the presence of at least one tuple having the same Personnel Number in the relation named QUAL. So when a query asking for personal details of the person also wishes

to find out whether he has any civil qualifications, and if so, what, the relation NAME being searched for the other details is also able to provide the answer to the first part of this query by indicating whether the person has any civil qualifications or not. In the case of a value of 'N' in this domain, the QUAL relation need not be searched at all. This helps in reducing the time for fruitless searches and thus helps in quicker query translation. But if any bit position in this byte got corrupted, it will not be possible to get this query answered so easily and may even give rise to an erroneous answer.

So the following simple method was utilized which made it possible not only the detection of error but error correction as well. The domain is first checked to find out whether it contains a Y or a N. Any other character in the domain gives rise to an error message. Next, for a value of 'Y', the byte was replaced by a set of all 1's and for 'N' was replaced by zero. This requires no extra space for the same byte position is utilized. All the above was achieved through a routine called YNCON. Next when the data is retrieved, another routine named YNDECO checks whether the number of 1's in the byte is greater than four or not. For all combinations in the byte having 1's in five or more bit positions, 'Y' is assumed to be the domain value and substituted and for all the remaining combinations, 'N' is assumed to be the domain value and substituted. This simple yet effective procedure caters for

anything upto three errors in any of the bit positions in the byte. The possibility of a higher number of errors occuring is considered to be negligible.

3-2.4 General Considerations in the Validation: Next we should come to a relation by relation implementation of the data validation requirements and the procedures used in fulfilling them. We have attempted to code only the domains where it was deemed extremely necessary, as stated before. I think a short explanation in lieu of this step is due here.

Originally it was thought that each individual domain should be coded so that at least error detection, and if possible also error correction would be facilitated. But later thoughts provoked us to decide in favour of attempting to code as little as possible. It was determined that in the model of data base that we have chosen for implementation, the queries, the type of domains and in fact everything indicated that the manipulations required to be performed with the domains were very simple in nature, mostly involving transfer, comparison and input/output type of operations. It was also determined that it would take a lot of computer time to convert the ASCII input to binary (in the case of decimal digit input), to code them for providing error detection and correction capabilities and then for checking these functions during retrieval, to decode them and then finally to convert them back to ASCII for

printing. The saving in space due to conversion to binary was not appreciable and the error detection and correction capabilities provided were found to be too small a gain compared to the time required for such operations. So the idea of converting and then coding the numeric portion of the data was finally given up. Due to the difficulties faced in coding the decimal digits directly, this idea also was given up.

Instead of these methods, another method was frequently used in this implementation. This method was to compute the check sums of the numeric fields, sometimes position wise but generally digitwise and to append these check sums to whatever positions within the tuple found to be most appropriate (mostly these were appended right at the end of the tuple). This procedure was followed whenever fresh tuples having a large number of numeric fields was being stored. Next, whenever this data was to be retrieved from the relation for subsequent manipulation, the check sums can be recomputed and composed with the previous value, and discrepancy giving rise to an error condition, signified by the print out of an error message followed by the print out of the offending record. This method appeared to provide a fair measure of error detection capability, though no error correction capability can be claimed.

3-2.5 Relation by Relation Analysis of the Implementation: Now we come to an actual relation by relation description of the data validation procedures implemented. In the first relation, named ARM in the previous chapter, we found that the corps domain was given as a two-digit decimal number. But for reasons of convenience in query translation, as desired by the person dealing with that part of the implementation, this domain was converted to binary and stored in one byte by a routine called ARMCON. This is unique to this domain, as it has been already explained that no other domain was converted to binary. The Personnel Number domain was suitably checked and coded. The domain of Rank does not merit any coding or validation. The only other domain in this relation, namely Time Scale, was a Y/N type of domain and was suitably validated and coded.

In the next relation Name, the Personnel Number, domain was validated and coded. The next two domains of Name-S and N Name-F were left alone after primary edit. In the Date-of-Birth domain, the year portion was checked for plausibility by means of a routine called YRLIM. The next two domains of Marital-Status and SC/ST are Y/N type of domains and were checked and converted. The next three domains were left alone after primary edit. All the next eight domains are of the Y/N type and were converted. The next three domains are again of a type that required no validation after primary edit. Since the number of numeric fields in this relation was small

Next relation QUAL is the first one which does not have a single domain as the primary key. Here a combination of the Personnel Number and the Civil-Degree-Code satisfies the constraint for uniqueness and was chosen as the primary key. This was validated after primary edit. The Personnel Number was then suitably coded. The year-passed-out domain was tested for plausibility using YRLIM. None of the other domains give rise to the possibility of validation and so were left alone after primary edit. But since all the domains are of numeric type, both the digitwise and positionwise checksums were computed and appended. This helps in detecting an error condition whenever it occurs in any of the actual domains but when an error has occured in any of the checksum domains (but not in both) the error is corrected internally so that processing can continue undisturbed. The checksumming was achieved by means of the routine QALCOD.

The next relation LANGI also has a combination of two domains Personnel Number and the Language-Code as the primary key. This is validated after primary edit for the property of uniqueness. The fourth field is of Y/N type and is converted by YNCON.

In the next relation LANGF, we again have a combination of the Personnel Number and Language-code domains as the primary key. This is validated after primary edit. The fourth domain is the Year-Passed which is checked for plausibility through YRLIM. All the other domains besides that of

Personnel Number being purely numeric, the positionwise check sum was computed and appended to the end of the tuple by the routine FLCON. It performed the additional duty of checking for the plausibility of the third domain, namely the Highest-Exam.-Passed-Code.

The next relation in our list is PQEAL. It again has a combination of two domains, the Personnel Number and the Prof-Course-Code as the primary key. After primary edit, the primary key validation is carried out. Next the Grades domain is checked for the correct values as only four types of grades are allowed. The Year-Passed is next checked for plausibility through YRLIM.

In our next relation MED, Personnel Number satisties the property of uniqueness and so was chosen as the primary key. It was validated for the above constraint after primary edit. Next the Personnel Number domain was coded. Finally the control was passed to the routine MEDCOD. This routine computed the digitwise checksum of each of the S,H,A,P and E blocks and appended them after their respective blocks. It at the same time computed the digitwise checksums of the different domain types, P,M.Y and V and appended these successively in the above order at the end of the tuple. This greater amount of trouble taken in this particular relation can be justified on the ground that any wrong information in this relation could cause premature retirement of an able-bodied officer as also the extension of service for some other person physically unfit for his job.

Next relation UNIT also has Personnel Number satisfying the property of uniqueness and so chosen as the primary key and validated after primary edit. Next comes the SUS No., a seven-digit decimal code which was left alone after primary edit. The next domain of the command-region was validated out of a fixed set of character strings by the routine COMCHK. The next domain of Date-of-TOS was validated for the plausibility of the year portion of it by YRLIM. The next domain was left alone. The next two domains are of the Y/N type and were properly coded after validation by YNCON. The next domain of subs-rank was checked against the present-rank domain for validity. The year portions of the dates of subs-rank and present-rank were again checked for plausibility by YRLIM. The last two domains were again of the Y/N type and were suitably converted.

The next relation of MINST has the combination of Personnel Number and Institute-Code domains as the primary key. After primary edit, validation of the primary key was carried out. There is only one other domain in this relation and it requires no other validation than primary edit.

The relation REMP which comes next has unique tuples corresponding to personnel number which corresondingly could be taken to act as the primary key. It was validated after primary edit. The next domain of the new-personnel number was validated and converted, again by PNCON. The year value in the Date-of-Retirement (the next domain) was checked for plausibility by YRLIM, as also the year portion of the Date-of-

Reemployment, the last domain of this relation. The other domains were passed over after primary edit.

The last relation in our present structure was DECOR, which again does not have unique personnel numbers for the tuples and a combination of this domain and the award-code domain was chosen as the primary key. This was validated after primary edit. The next domain of year was checked for plausibility by YRLIM. The last domain of command-region was validated through the routine COMCHK.

We had been discussing data validation with respect to the initial phase of data base building. Two other routines were heavily utilized in this phase, one for writing the data onto the disk and the other for selecting the area where the data is to be placed, i.e., for choosing the cylinder, surface and sector numbers and incrementing them properly after each write operation. The first is called WRITE and the second is called ARSLCT. Another routine SET is used at the beginning of the whole operation for the purpose of initialisation, before any write operation is attempted.

3-3. VALIDATION DURING RETRIEVAL

The next phase of data validation is done when a tuple is retrieved from the data base for the purpose of manipulation or report generation. Here again, for a particular query, a particular relation is chosen which contains the answer to the query. This may involve several intermediate stages concerning several other relations, choosing different sets of domains and

so on from different relations until finally all the conditions specified in the query are fully satisfied. This also requires searching of these relations to find matches for the primary keys specified. This portion of the project falls under query translation taken up by another member of the project group. In this phase he is supposed to utilise the routines developed for the purpose of data validation.

The validation requirements in this phase were mainly concerned with the validation of the checksums to find out whether any portion of the data had been corrupted and listing out the error message together with the offending tuple in the case of its occurrence. Other possible validations were that of the primary key, the primary edit to satisfy that character type had not changed and the validation of some of the inter-field relationships. But these last portions are all optional, since we may assume that in case the checksums are validated, they are unlikely to occur and the other possible error, namely the duplication of primary keys is also unlikely in case the uniqueness constraint had been satisfied initially when building the data base.

The routines used at this stage for the checksum validations will now be described. The routine COMDEC calculates the checksums of the date-fields and the other numeric fields in the COMM relation separately. It also sums these two checksums and then compares the three values with the three values computed earlier during the building phase and appended at the tail of the tuple. If either the first or the second

mismatches and the third is also found to be wrong, then error is indicated. But if either the first or the second mismatch but the third is found to be alright, then it can be safely assumed that only the appropriate checksum domain has been corrupted and this is replaced by the correct value and the data presented for manipulation without any error being indicated.

In the relation QUAL, the subroutine QALDEC performs a similar function. As described earlier, in this relation the digitwise and positionwise checksums of the numeric fields were computed and appended at the end. These are now re-computed and compared. Mismatch for both values indicates that error had occurred. But if only one value mismatches, then that checksum only has been corrupted and after its replacement by the correct value, processing of the tuple can continue.

The routine FLDEC validates the relation LANGF. Here again the positionwise checksum is recomputed and compared with the previous value, any discrepancy giving rise to an error message.

The routine MEDVAL is used to validate the relation MED. As discussed earlier, major efforts were put into compute the digitwise checksums in this relation. The checksums for each of the five blocks were computed and appended at the end of that block while the checksum for each type of field was also computed separately and these were appended in the order of appearance of the fields at the end of the tuple. These are now recomputed and comparisons made. Any error in the actual

data items would give rise to discrepancy in two of the checksum domains, one belonging to that block and the other to that type of field. A cross checking ensures that this has actually occurred and then the error can be pinpointed as having occured in a particular type of field in a particular block. If only one checksum was found to be mismatched then that one was assumed to be corrupted and was replaced without indicating any error. This extra effort in this relation can be justified on the ground that the relation has numeric fields of total length 35 consecutive bytes and unless the exact error location is pin pointed, it becomes a very laborious process to try to locate it.

This brings our discussion of the second phase of data validation in our data base project to an end.

3-4. VALIDATION AFTER UPDATE

In the third phase, namely that during the storage of the data after manipulation, the requirements for validation are generally the same as in the first stage, as discussed earlier. The data may be subjected to primary edit. Next comes primary key validation. This assumes greater significance in that a constituent of the primary key could have been operated on, may be to correct an initial mistake, giving rise to the possibility of primary key duplication. Also the plausibility and limit checks need to be done again as an altered data item may violate these constraints. Similarly the interfield relationship checks must also be made. If some of the Y/N type of fields have been replaced these need to be converted

again. Finally the new checksums must be computed to replace the previous values. As is obviously true, the routines developed to serve in the data base building phase suffice in this phase also, except when domains are deleted or added or domain types changed or interchanged. This, of course, changes the structure of the relation and of the whole data base as such and is not the topic of discussion here.

3-5. VALIDATION DURING REPORT GENERATION

In the last phase of our work, namely during the phase of report generation, the validation requirements are similar to the second phase of data retrieval, for data must first be retrieved before report generation can be attempted. After the validation during the retrieval of the data, the following steps must be followed. The Y/N type of domains are decoded to their proper values by the YNDECO routine. Next the checksums are to be removed and packing of the data affected if necessary. Finally the Personnel Number must be relieved of the extra digit added to it during the coding process. After this the data is presented for output.

This completes the discussion of the data validation requirements and the routines utilized to achieve these for the data base project.

4. SOME DETAILS OF THE DATA VALIDATION ROUTINES

4-1. INTRODUCTION

This chapter describes the various algorithms developed for the actual validation of the data. The routines described here are mentioned in their particular places of usage in the previous chapter and the detailed listing of the program which contains these routines and uses them is appended at the end of this write up. Since the detailed listing is given, only a few of the algorithms will be described in a step by step manner, the rest of the algorithms would not be specified in any great detail.

Some general observations are appropriate here. The TDC-316 on which the system was implemented had 8 registers in all, being numbered from 0 to 7. Of these the first two, namely, 0 and 1 are the location counter and stack pointer registers respectively, so these were not used for other purposes. The other registers were named R2 to R7 respectively and were widely used. Of these R2 was specifically used in subroutine calls, though it was also used otherwise. The buffer area BF3 was generally used as the common input/output area for the main routine as well as in nearly all subroutines.

4-2. GENERAL ROUTINE DESCRIPTION

We will first describe the general routine that is used for initially validating the data and putting it in the data base. It was developed in a manner suitable for our data

base and at present cators for the twelve relations built up but may be extended, to accomodate any further relations developed later, with ease. There is, of course, no general algorithm as such. It involves first reading in the data about the relation to be built up, namely the relation number, its present format, the number of tuples to be placed in this relation, the limiting values of the years and such other plausibility limits and finally the starting address, i.e., the cylinder, surface and sector numbers from which the relation should be placed on the disk. The format specifications are checked and coded properly by PEFORM as stated earlier, NOFREC checks whether the number of tuples to be placed is a positive, non-zero integer or not. Next the relation number is decoded and jump made to the particular section of the general routine dealing with that relation. These sections check the various formats, primary keys, various interfield relationships and plausibility limits, calling various subroutines in the process, compute and append checksums where appropriate (as discussed in Chapter 3) and then finally selects the proper area on the disk and places it there for the later purpose of data base BUILD function, being developed by a coworker, as stated earlier.

4-3. DESCRIPTION OF OTHER ROUTINES USED IN THE FIRST PHASE

Now we will describe the subroutines used in the building phase of the data base. The routines would be described in the order in which they appear in the listing of the programme.

<u>Subroutine PEFORM</u>: This routine makes use of two buffers, a one word buffer called FMT1 as temporary storage for manipulation and assembly of each of the domain formats and another larger buffer FMT where each domain format is placed in successive word locations.

<u>Algorithm</u>: The format card is read in and the data stored digitwise as an array represented here by F(I).

Step 1: Set $N \leftarrow 0$, $I \leftarrow 1$, $K \leftarrow 80$.

Step 2: Compare F(I) to 'blank'. If equal go to Step 8.

Step 3: Compare F(I) with 1,2,3 and 4 successively and place octal '0', '20', '40' and '60' respectively for a match with these in the upper byte of FMT1. If no match is found, print error message and go out.

Step 4: Convert F(I+1) and F(I+2) considering them as the 1st and 2nd digits of a decimal number and store the binary result in the lower byte of FMT1.

Step 5: $FMT(N+1) \leftarrow FMT1$.

Step 6: $K \leftarrow K - 3$, if $K \not> 3$ go to Step 8.

Step 7: $I \leftarrow I+3$, $N \leftarrow N+1$, go to Step 2.

Step 8: If $F(I+3) \neq$ 'blank', then read in next card, set $I \leftarrow 1$ and $K \leftarrow 80$ and go back to Step 2; else $N \leftarrow N+1$, $FMT(1) \leftarrow N$, stop.

<u>Subroutine NOFREC</u>: This subroutine reads in the number of records to be placed in the relation, converts it to binary, confirms that it is a nonzero positive integer and then reduces its value by 1 and returns control to the main routine. Its

description is omitted.

<u>Subroutine PECHEK</u>: This subroutine reads in the tuple and checks it against the previously specified formats, stored in array FMT. The data is read into buffer BF3 which will be represented here by an array BF.

<u>Algorithm:</u>

Step 1: Check if number of fields (stored inFMT(1) is zero, if so indicate error and exit.

Step 2: I ← 2, N ← FMT(L), J ← 1

Step 3: Transfer lower byte of FMT(I) to R4 and upper byte to R5.

Step 4: Compare R4 with octal '0'. If equal go to Step 5.

Compare R4 with octal '20'. If equal go to Step 7.

Compare R4 with octal '40'. If equal go to Step 9.

Compare R4 with octal '60'. If equal go to Step 11.

Print error message and exit.

Step 5: Check if BF(J) lies between octal '60' and '71' else indicate error and exit.

Step 6: J ← J+1, R5 ← R5-1,

If R5 > 0 go to Step 5 else go to Step 12.

Step 7: Check if BF(J) lies between Octal '0' and '1' else indicate error and exit.

Step 8: J ← J+1, R5 ← R5 - 1. If R5 > 0 go to Step 7 else go to Step 12.

Step 9: Check if BF(J) lies between octal '101' and '132' or equal to 'blank', else indicate error and exit.

Step 10: $J \leftarrow J + 1$, $R5 \leftarrow R5 - 1$. If $R5 > 0$, go to Step 9 else go to Step 12.

Step 11: $J \leftarrow J + R5$.

Step 12: $I \leftarrow I+3$, $N \leftarrow N-1$. If $N > 0$ go to Step 3 else stop.

<u>Subroutine PNCON:</u> This calls two further subroutine PNCV and PNDC. The first one checks the alphabetic portion of the Personnel No. string and verifies whether it belongs to the allowed set or not (in the latter case indicates error and exits). The second subroutine computes the sum of the digits of the decimal portion and then the sum of the digits of the sum and so on until a single digit is obtained and this is appended to the end of the string.

<u>Subroutine YNCON:</u> This subroutine takes as input a buffer area of one byte called YNBF, checks to find whether the contents are the equivalent of ASCII 'Y' or 'N', otherwise indicating error and exiting, then finally substitutes a string of all 1's for 'Y' and all '0's for 'N' in YNBF.

<u>Subroutine YRLIM:</u> This subroutine utilises three buffer areas of size one word each, YRUL to contain the upper limit of the year value, YRLL to contain the lower limit of the year value and YLBF to contain the actual year value whose check must be made. Algorithm is again pretty simple consisting of checking YLBF against YRUL first and then against YRLL, either value being contravened indicated by error message and exit.

<u>Subroutine ARSLCT</u>: This subroutine selects the area on the disk at which the next tuple will be placed. The starting address of the relation is given initially as input in the form of cyliner, surface and sector numbers. Next, whenever a tuple is placed on the disk, it increments the sector number and checks whether it has exceeded ten or not. If it has exceeded 10, it is replaced by 1 and the surface value is incremented by 1. In that case, the surface value is next checked to find out if it has exceeded 9. If so, the value in the cylinder field is incremented by 1 and the surface and sector numbers are replaced by 0 and 1 respectively.

<u>Subroutines for Disk I/O:</u> These consist of three parts, namely set, read and write, of which only two, set and write were useful for our purpose in this section. These routines were developed by a coworker and so would not be discussed here.

<u>Subroutine COMCHK:</u> This routine checks the command-region value for validity of that domain. If is a very simple routine which takes the value of the domain from a buffer called COMBF1 and compares it with a fixed set of values, all of which unmatched gives rise to an error message and exit.

<u>Subroutine PKCHEK:</u> This subroutine validates the primary key for duplication and hence is a very important routine for our work. It uses three one-word buffers called PKN, PKL and PKB respectively. Of these, PKN is used to contain the number of primary keys stored till then in the primary key buffer whose starting address is placed in PKB. PKL contains the

length of the primary key of the tuples of the relation being presently stored. The array F(I) is used here to represent the primary key buffer and BF is used to represent the input buffer.

Algorithm

Step 1: Set N ← PKN, I ← 1

Step 2: Check if N is zero, then go to Step 7

Step 3: Set J ← PKL, K ← 1

Step 4: Compare BF(K) with F(I). If unequal go to Step 6.

Step 5: K ← K+1, I ← I + 1, J ← J - 1. If J > 0, go to Step 4, else print error message and wait for restart. On restart go to Step 1.

Step 6: K ← K + J, I ← I + J, N ← N - 1. If N > 0, go to Step 3.

Step 7: Set J ← PKL, K ← 1

Step 8: F(I) ← BF(K), I ← I + 1, K ← K + 1, J ← J - 1. If J > 0, go to Step 8.

Step 9: PKN ← PKN + 1,
Stop.

Subroutine COMCOD: This subroutine computes the digitwise checksums of the date domains and the other numeric domains separately as well as their sum and append these to the end of the tuple in the relation COMM. One-word buffers CMBF2 and CMBF3 are used to indicate the positions in the tuple from which the input is to be taken and where the checksums are to be placed respectively.

Subroutine PQLCHK: This subroutine simply checks the grades in the relation PQUAL for validity. The number of allowed grades is a small set and it compares the input in the buffer PQLBF with these. A failure to match any of these gives an error message and sets a flag PQF which may be subsequently used in the main routine to determine whether to insert the tuple or not.

Subroutine ARMCON: This subroutine converts the input from Corps domain in buffer ARMBF1 from two byte ASCII to one byte binary and is of very elementary nature.

Subroutine FLCON: In this subroutine, we first check for plausibility of the level of exam passed. Next we calculate the positionwise checksum of the numeric domains and append it to the end of the tuple. At the same time the bytes of the tuple are transferred from the input buffer BF3 to the foreign language buffer FLBF.

Subroutine MEDCOD: This is one of the larger subroutines where, as stated earlier (in Chapter 3), we compute the sum of each individual type of domains P, M,Y and V, appending these to the end of the tuple, while at the same time the digitwise checksums of the blocks S,H,A,P and E are calculated and appended to the end of each block respectively. Two one-word buffers TEMP and TEMP1 are used for intermediate or temporary storage in the process. Other buffers used are one-word buffers P,M,Y and V named after the domains respectively.

Subroutine INCVAL: This is again a very simple subroutine that validates the Institute-of-Commissioning domain. The values in this domain can again be from a fixed set of alphabetic character strings and these are matched with the present domain value character by character, any discrepancy giving rise to an error message.

Subroutine QALCOD: This subroutine calculates both the positionwise and the digitwise checksums of the numeric fields in the relation QUAL and appends these at the end of tuple in that order respectively. It also transfers the input string from buffer BF3 to buffer QULBF to help in writing on the disk. It uses the address of the input string given in the buffer QLBF and utilises two one-word buffers QLTOT and QLTOT1 for calculating the checksums.

This completes our discussion of the routines used in the first phase.

4-4. DESCRIPTION OF THE OTHER ROUTINES

These routines were developed to decode or check whatever data were coded or checksummed in the first phase. These routines are to be utilised in the data retrieval and related phases by the person developing that part of the DBMS and so no general routine with a calling sequence for these was developed.

<u>Subroutine QALVAL</u>: This subroutine is used to recompute the positionwise and digitwise checksums of the numeric domains in the relation QUAL and compare them with the previously computed values stored at the end of the tuple. In the case that one of the checksums only mismatches, that is assumed to be in error and is replaced by the new value without indicating error. But if both the checksums mismatch then some of the numeric domains may be in error and this is indicated by an error message with a print out of the offending tuple.

<u>Subroutine FLDEC:</u> This subroutine recomputes the position wise checksum of the numeric domains in the relation LANGF and compares it with the previously calculated and stored value, any discrepancy giving rise to an error message.

<u>Subroutine YNDECO</u>: This subroutine checks the contents of a buffer YNBUF for the number of 1's present and converts it accordingly to 'Y' or 'N'. A is a one-byte buffer which is used to make the comparisons and it has only its first bit position a 1 and the rest 0s. Let us assume that the binary string is present in array BF.

<u>Algorithm</u>

Step 1: $I \leftarrow 8$, $J \leftarrow 0$

Step 2: Compare BF(I) with 1. If equal, then $J \leftarrow J + 1$

Step 3: $I \leftarrow I - 1$. If $I > 0$, go to Step 2.

Step 4: If $J > 4$, then YNBF = 'Y' else YNBF = 'N'.

Stop.

Subroutine COMDEC: This subroutine computes the digitwise sum of the date domains and of the other numeric domains in the relation COMM separately as well as computing the sum of the two. These are then compared with the previously computed and stored values. Any single mismatch is assumed to be in the checksums themselves and these are replaced. In all other cases an error message followed by a printout of the tuple occurs.

Subroutine MEDVAL: This subroutine recomputes the checksums calculated by the MEDCOD routine and compares them with the previous values. It first goes on a block by block basis, comparing the block checksum on reaching the end of each block. If any discrepancy occurs, the corresponding block error flag is set. When it reaches the end of the last block, it checks whether any of the flags are set; otherwise it just replaces the domain checksums by the newly computed values and exits. But if any flag is set, it starts checking the domain checksums. If none of them mismatch, it replaces the block checksums by the new value and exits. When both some block and domain type checksums are found to be mismatched, it prints the offending tuple after a message indicating the exact error location in terms of the block and domain.

Subroutine BINASC: This subroutine converts a one byte binary number to a 3-digit decimal number and is utilised with many of the validate routines of this section since otherwise the tuples printed will contain many odd characters in the checksum fields which are in binary form.

This completes our discussion of the routines used to validate the data in our data base.

5. CONCLUSION

As stated early in this work, not much work has been done in the field of data validation in a data base. So this field needs to be investigated further and presently some interest seems to have been created in it.

It was also stated in the first chapter that our aim was to implement a relational model of data base management system to investigate its claims of superiority to other models. This present work does not involve much of the relational model except where the structure was built up, i.e., in the work of normalizing the data base. It was found that the data validation requirements were not much different from what they would have been in the other models. But one thing was found out, that this model helped in determining the exact requirements and in accessing the portions of the data required to be validated very easily, due to the flat file sort of structure of the relations. This simplicity was very much evident in comparison with the chain-link determination in the network model and the tree traversals that would have been required in the hierarchical approach. The validation of primary keys and other similar single domains was specially made easier by this structure.

—

BIBLIOGRAPHY

1. Codd, E.F., 'A relational model of data for large shared data banks', Comm. ACM, 13, 6 (June 1970), pp. 377-397.
2. Codd, E.F., 'Relational completeness of data base sub-languages', Courrent Computer Science Symposium 6, 'Data Base Systems', New York, May 1971, pp. 65-98.
3. Codd, E.F., 'Further normalization of the data base relational model', Courant Computer Science Symposia 6, 'Data Base Systems', New York, May 1971, pp. 33-64.
4. Codd, E.F., 'Relational Algebra'. Courant Computer Science Symposia 6, 'Data Base Systems', New York, May 1971.
5. Date, C.J., 'An Introduction to Data Base Systems', Addison-Wesley, 1975.
6. Martin, J., 'Computer Data Base Organization', Prentice-Hall, Inc., 1975.
7. Sibley, E.H.., 'Guest Editor's Introduction : The Development of Data Base Technology', ACM Computing Surveys, March 1976, Vol. 8, Number 1, pp. 1-6.
8. Lott, R.W., 'Basic Data Processing', 2nd Ed. Prentice-Hall, Inc., Englewood Cliffs, 1971.
9. Gotlieb, C.C., and Hume, J.N.P., 'High-Speed Data Processing', McGraw-Hill, New York, 1958.
10. Wilkes, M.V., 'On Preserving the Integrity of Data Bases', The Computer Journal, Vol. 15, No. 3, (August 1972).

11. Bayer, R., 'On the Integrity of Data Bases and Resource Locking', Technische Universitat Munchen - Data Base Systems, Proceedings of 5th Information Symposium, IBM Germany, Bad Homburg, v.d.H., September 1975.

12. Class Notes of Professor Rajaraman in the Course CS 650, 'Design of Information Systems' at IIT Kanpur.

13. Fry, James, P., and Sibley, E.H., 'Evolution of Data-Base Management Systems', ACM Computing Surveys, March 1976, Vol. 8, No. 1, pp. 7-72.

14. Emery, Glyn, 'Electronic Data Processing', American Elsevier, New York (1969, c1968).

15. Chamberlin, D.D., 'Relational Data-Base Management Systems', ACM Computing Surveys, March 1976, Vol. 8, No. 1, pp.43-66.

```
      ;
      ;
      ;BEGINNING OF THE GENERAL ROUTINE
      ; THE MAIN ROUTINE USED TO VALIDATE THE DATA AT INPUT AND PLACE IT ON
      ; THE DISK IN THE FINAL FORM OF THE 12 RELATIONS
      ;
 ; DEFINE CONSTANTS AND REGISTERS
      ;
      .=50000
      R2=%2
      R3=%3
      R4=%4
      R5=%5
      R6=%6
      R7=%7
      N2=2
      N3=3
      N4=4
      N5=5
      MQ=177736
      RESET
      INIT  N2,BF1 ;INITIALIZE THE READER
      INIT  N3,BF2 ; INITIALIZE THE PRINTER
      INIT  N4,BF7 ; INITIALIZE THE TELETYPE
      INIT  N5,BF14 ; INITIALIZE THE KEYBOARD
      JMS   R4,SET
      TSR   #40000,PKB ; STARTING ADDRESS OF THE PRIMARY KEY BUFFER
PL21: SWRITE  N4,PBF5
 L43: WAITR  N4,L43
      SREAD  N5,PBF1
 PL1: WAITR  N5,PL1
      BTSR  PBF1+6,PBF2+35
      BTSR  PBF1+7,PBF2+36
      SWRITE  N3,PBF2
PL66: WAITR  N3,PL66
      JMS   R2,NOFREC
      JMS   R2,PEFORM
      STOP
      SREAD  N2,FRBF
PL23: WAITR  N2,PL23
  ; FOLLOWING SECTION DETERMINES THE DSTARTING DISK ADDRESS
      SREAD  N2,DASBF
 DL1: WAITR  N2,DL1
      CLR   R2
      TSR   #12,MQ
      BTSR  DASBF+6,R2
      SUB   #60,R2
      MPY   R2,R2
      TSR   MQ,R2
      CLR   R3
      TSR   #12,MQ
      BTSR  DASBF+7,R3
      SUB   #60,R3
      MPY   R3,R3
      ADD   MQ,R2
      CLR   R3
```

```
        BRZS  PL3
        SUB   #4,R3
        BRZC  PL2
        JMP   PL4
 PBF2:  WORD  33,0,33
        BCI   % BUILDING RELATION NO, %
        BYTE  0,0,15,12
        EVEN
DASBF:  WORD  6,0,6
        .=.+6
 BF14:  WORD  1
PBF11:  WORD  35,0,35
        BCI   % RELATION DOES NOT EXIST %
        BYTE  15,12
        EVEN
; JUMP TO THE DIFFERENT SECTIONS FOR THE DIFFERENT RELATIONS.
; THE JUMPS ARE ORDERED IN THE REVERSE ORDER TO THE NO. OF THE RELATIONS.
; SC FOR JUMPS TO ANY NEW SECTIONS THESE SHOULD BE ADDED TO THE TOP
  PL3:  JMP   PL3(R3)
        JMP   PL20
        JMP   PL17
        JMP   PL16
        JMP   PL15
        JMP   PL14
        JMP   PL13
        JMP   PL12
        JMP   PL11
        JMP   PL10
        JMP   PL7
        JMP   PL6
        JMP   PL5
 PBF1:  WORD  3,0,3
        .=.+4
 FREF:  WORD  120,0,120
        .=.+120
  PL4:  SWRITE  N4,PBF11
 PL22:  WAITR   N4,PL22
        STOP
        JMP   PL21
   WR:  WORD  111,0,111
        BYTE  15,12
       BCI %WHAT IS WORDLENGTH OF RECORD%
       BCI %(AS TWO DIGIT DECIMAL INTEGER ON KBD)%
        BYTE 15,12
        EVEN
   RL:  WORD  3,0,3
        .=.+4
 DHBF:  WORD  0
        .=.+400
PDRBF:  WORD  0
 PBF5:  WORD  50,0,50
        BYTE  15,12
        BCI %GIVE REL. NO.(AS TWO-DIGIT INTEGER)%
        BYTE  15,12
        EVEN
 NRPS:  WORD  0
```

```
      BTSR  DASBF+10,R3
      SUB   #60,R3
      ADD   R3,R2
      TSR   R2,CYL
      CLR   R2
      BTSR  DASBF+11,R2
      SUB   #60,R2
      TSR   #12,MQ
      MPY   R2,R2
      TSR   MQ,R2
      TSR   #12,MQ
      CLR   R3
      BTSR  DASBF+12,R3
      SUB   #60,R3
      MPY   R3,R3
      ADD   MQ,R2
      CLR   R3
      BTSR  DASBF+13,R3
      SUB   #60,R3
      ADD   R3,R2
      TSR   R2,DIS
      SWRITE  N4,WR
  L3: WAITR  N4,L3
      SREAD  N5,RL
  L16: WAITR  N5,L16
      CLR   R2
      CLR   R3
      BTSR  RL+6,R2
      SUB   #60,R2
      TSR   #12,MQ
      MPY   R2,R2
      TSR   MQ,R2
      BTSR  RL+7,R3
      SUB   #60,R3
      ADD   R3,R2
      TSR   #200,MQ
      CLR   R3
      DIV   R2,R3
      TSR   MQ,NRPS
      TSR   #DRBF,PDRBF
      TSR   NRPS,NRPS1
      STOP
; FOLLOWING SECTION DETERMINES THE NO. OF THE RELATIO FOR JUMP TO THAT SECTION.
      TSR   #60,R3 ; THIS VALUE TO BE INCREMENTED BY 4 FOR EACH NEW RELATION
      ;  ADDED TO THE DATABASE
      CLR   R2
      CLR   R4
      BTSR  PBF1+6,R2
      SUB   #60,R2
      TSR   #12,MQ
      MPY   R2,R2
      TSR   MQ,R2
      BTSR  PBF1+7,R4
      SUB   #60,R4
      ADD   R4,R2
  PL2: DEC   R2
```

```
        JMS   R2,YNCON
        BTSR   YNBF,NAMEBF+66
        BTSR   BF3+74,YNBF
        JMS   R2,YNCON
        BTSR   YNBF,NAMEBF+67
        TSR   #5,R2
PL26:   BTSR   BF3+74(R2),NAMEBF+67(R2)
        DEC   R2
        BRZC   PL26
        BTSR   BF3+102,YNBF
        JMS   R2,YNCON
        BTSR   YNBF,NAMEBF+75
        BTSR   BF3+103,YNBF
        JMS   R2,YNCON
        BTSR   YNBF,NAMEBF+76
        BTSR   BF3+104,YNBF
        JMS   R2,YNCON
        BTSR   YNBF,NAMEBF+77
        BTSR   BF3+105,YNBF
        JMS   R2,YNCON
        BTSR   YNBF,NAMEBF+100
        BTSR   BF3+106,YNBF
        JMS   R2,YNCON
        BTSR   YNBF,NAMEBF+101
        BTSR   BF3+107,YNBF
        JMS   R2,YNCON
        BTSR   YNBF,NAMEBF+102
        BTSR   BF3+110,YNBF
        JMS   R2,YNCON
        BTSR   YNBF,NAMEBF+103
        BTSR   BF3+111,YNBF
        JMS   R2,YNCON
        BTSR   YNBF,NAMEBF+104
        BTSR   BF3+112,NAMEBF+105
        BTSR   BF3+113,NAMEBF+106
        TSR   PDRBF,R2
        TSR   #107,R4
        TSR   #NAMEBF,R3
 L31:   BTSR   (R3)+,(R2)+
        DEC   R4
        BRZC   L31
        TSR   R2,PDRBF
        DEC   NRPS
        BRZC   L32
        JMS   R2,DWR
 L32:   DEC   C
        BRSC   PL60
        CMP   #0,NRPS
        BRZS   L33
        JMS   R2,DWR
 L33:   TSR   NRPS1,NRPS
        JMP   PL21
PL60:   JMP   PL40
NAMEBF: WORD   0
        .=.+110
; BEGINNING OF SECTION FOR RELATION ' 3'
```

```
NRPS1: WORD  0
; BEGINNING OF SECTION FOR RELATION ' 1'
 PL5: TSR   #BF3+6,PNBF
      TSR   #ARMBF,PNBUF
      CLR   PKN
      TSR   #11,PKL
PL51: JMS   R2,PECHEK
      JMS   R2,PKCHEK
      JMS   R2,PNCON
      BTSR  BF3+15,ARMBF1
      BTSR  BF3+16,ARMBF1+1
      JMS   R2,ARMCON
      BTSR  ARMBF1,ARMBF+10
      BTSR  BF3+17,ARMBF+11
      BTSR  BF3+20,YNBF
      JMS   R2,YNCON
      BTSR  YNBF,ARMBF+12
      TSR   PDRBF,R2
      TSR   #ARMBF,R3
      TSR   #15,R4
 L26: BTSR  (R3)+,(R2)+
      DEC   R4
      BRZC  L26
      TSR   R2,PDRBF
      DEC   NRPS
      BRZC  L27
      JMS   R2,DWR
 L27: DEC   C
      BRSC  PL51
      CMP   #0,NRPS
      BRZS  L30
      JMS   R2,DWR
 L30: TSR   NRPS1,NRPS
      JMP   PL21
; BEGINNING OF SECTION FOR RELATION ' 2'
 PL6: TSR   #BF3+6,PNBF
      TSR   #NAMEBF,PNBUF
      TSR   FRBF+6,YRUL
      TSR   FRBF+10,YRLL
      CLR   PKN
      TSR   #7,PKL
PL40: JMS   R2,PECHEK
      JMS   R2,PKCHEK
      JMS   R2,PNCON
      TSR   #50,R2
PL24: BTSR  BF3+14(R2),NAMEBF+7(R2)
      DEC   R2
      BRZC  PL24
      BTSR  BF3+65,YLBF
      BTSR  BF3+66,YLBF+1
      JMS   R2,YRLIM
      TSR   #6,R2
PL25: BTSR  BF3+64(R2),NAMEBF+57(R2)
      DEC   R2
      BRZC  PL25
      BTSR  BF3+73,YNBF
```

```
 L35: DEC   C
      BRSC  PL41
      CMP   #0,NRPS
      BRZS  L36
      JMS   R2,DWR
 L36: TSR   NRPS1,NRPS
      JMP   PL21
; BEGINNING OF SECTION FOR RELATION ' 4'
PL10: TSR   #BF3+6,PNBF
      TSR   #QULBF,PNBUF
      CLR   PKN
      TSR   #11,PKL
PL42: JMS   R2,PECHEK
      JMS   R2,PKCHEK
      JMS   R2,PNCON
      TSR   FRBF+6,YRUL
?     TSR   FRBF+10,YRLL
      TSR   BF3+24,YLBF
      JMS   R2,YRLIM
      TSR   #BF3+15,QLBF
      JMS   R2,QALCOD
      TSR   PDRBF,R2
      TSR   #24,R4
      TSR   #QULBF,R3
 L37: BTSR  (R3)+,(R2)+
      DEC   R4
      BRZC  L37
      TSR   R2,PDRBF
      DEC   NRPS
      BRZC  L46
      JMS   R2,DWR
 L46: DEC   C
      BRSC  PL42
      CMP   #0,NRPS
      BRZS  L47
      JMS   R2,DWR
 L47: TSR   NRPS1,NRPS
      JMP   PL21
;
; BEGINNING OF SECTION FOR RELATION ' 5'
PL11: TSR   #BF3+6,PNBF
      TSR   #LIBF,PNBUF
      CLR   PKN
      TSR   #11,PKL
PL43: JMS   R2,PECHEK
      JMS   R2,PKCHEK
      JMS   R2,PNCON
      BTSR  BF3+20,YNBF
      JMS   R2,YNCON
      BTSR  YNBF,LIBF+13
      BTSR  BF3+15,LIBF+10
      BTSR  BF3+16,LIBF+11
      BTSR  BF3+17,LIBF+12
      TSR   PDRBF,R2
      TSR   #13,R4
      TSR   #LIBF,R3
```

```
PL7:  TSR   #BF3+6,PNBF
      TSR   #COMBF,PNBUF
      CLR   PKN
      TSR   #7,PKL
PL41: JMS   R2,PECHEK
      JMS   R2,PKCHEK
      JMS   R2,PNCON
      TSR   FRBF+6,YRUL
      TSR   FRBF+10,YRLL
     BTSR   BF3+15,YLBF
     BTSR   BF3+16,YLBF+1
      JMS   R2,YRLIM
      TSR   FRBF+12,YRUL
      TSR   FRBF+14,YRLL
     BTSR   BF3+23,YLBF
     BTSR   BF3+24,YLBF+1
      JMS   R2,YRLIM
      TSR   FRBF+16,YRLL
      TSR   FRBF+20,YRLL
     BTSR   BF3+31,YLBF
     BTSR   BF3+32,YLBF+1
      JMS   R2,YRLIM
      TSR   FRBF+22,YRUL
      TSR   FRBF+24,YRLL
      BTSR  BF3+40,YLBF
      BTSR  BF3+41,YLBF+1
      JMS   R2,YRLIM
      TSR   FRBF+26,YRUL
      TSR   FRBF+30,YRLL
      BTSR  BF3+44,YLBF
      BTSR  BF3+45,YLBF+1
      JMS   R2,YRLIM
      TSR   #COMBF+63,CMBF3
      TSR   #BF3+15,CMBF2
      JMS   R2,COMCOD
      TSR   #BF3+50,CMBUF
      JMS   R2,INCVAL
      TSR   #BF3+15,R2
      TSR   #COMBF+10,R3
      TSR   #45,R4
PL27: BTSR  (R2)+,(R3)+
      DEC   R4
      BRZC  PL27
      BTSR  COMBF+46,YNBF
      JMS   R2,YNCON
      BTSR  YNBF,COMBF+46
      TSR   PDRBF,R2
      TSR   #100,R4
      TSR   #COMBF,R3
L34:  BTSR  (R3)+,(R2)+
      DEC   R4
      BRZC  L34
      TSR   R2,PDRBF
      DEC   NRPS
      BRZC  L35
      JMS   R2,DWR
```

```
 L50: BTSR  (R3)+,(R2)+
      DEC   R4
      BRZC  L50
      TSR   R2,PDRBF
      DEC   NRPS
      BRZC  L51
      JMS   R2,DWR
 L51: DEC   C
      BRSC  PL43
      CMP   #0,NRPS
      BRZS  L52
      JMS   R2,DWR
 L52: TSR   NRPS1,NRPS
      JMP   PL21
LIBF: WORD  0
      .=.+12
; BEGINNING OF SECTION FOR RELATION ' 6'
PL12: TSR   #BF3+6,PNBF
      TSR   #FLBF,PNBUF
      CLR   PKN
      TSR   #11,PKL
PL44: JMS   R2,PECHEK
      JMS   R2,PKCHEK
      JMS   R2,PNCON
      TSR   FRBF+6,YRUL
      TSR   FRBF+10,YRLL
      TSR   BF3+20,YLBF
      JMS   R2,YRLIM
      JMS   R2,FLCON
      TSR   PDRBF,R2
      TSR   #16,R4
      TSR   #FLBF,R3
 L53: BTSR  (R3)+,(R2)+
      DEC   R4
      BRZC  L53
      TSR   R2,PDRBF
      DEC   NRPS
      BRZC  L54
      JMS   R2,DWR
 L54: DEC   C
      BRSC  PL44
      CMP   #0,NRPS
      BRZS  L55
      JMS   R2,DWR
 L55: TSR   NRPS1,NRPS
      JMP   PL21
; BEGINNING OF SECTION FOR RELATION ' 7'
PL13: TSR   #BF3+6,PNBF
      TSR   #PQBF,PNBUF
      TSR   FRBF+6,YRUL
      TSR   FRBF+10,YRLL
      CLR   PKN
      TSR   #11,PKL
PL45: JMS   R2,PECHEK
      JMS   R2,PKCHEK
      JMS   R2,PNCON
```

```
        TSR   BF3+20,YLBF
        JMS   R2,YRLIM
        BTSR  BF3+17,PGLBF
        JMS   R2,PGLCHK
        TSR   #BF3+15,R2
        TSR   #PQBF+10,R3
        TSR   #5,R4
PL30:   BTSR  (R2)+,(R3)+
        DEC   R4
        BRZC  PL30
        TSR   PDRBF,R2
        TSR   #16,R4
        TSR   #PQBF,R3
 L56:   BTSR  (R3)+,(R2)+
        DEC   R4
        BRZC  L56
        TSR   R2,PDRBF
        DEC   NRPS
        BRZC  L57
        JMS   R2,DWR
 L57:   DEC   C
        BRSC  PL45
        CMP   #0,NRPS
        BRZS  L60
        JMS   R2,DWR
 L60:   TSR   NRPS1,NRPS
        JMP   PL21
; BEGINNING OF SECTION FOR RELATION ' 8'
PL14:   TSR   #BF3+6,PNBF
        TSR   #MCBF,PNBUF
        CLR   PKN
        TSR   #7,PKL
PL46:   JMS   R2,PECHEK
        JMS   R2,PKCHEK
        JMS   R2,PNCON
        JMS   R2,MEDCOD
        TSR   PDRBF,R2
        TSR   #64,R4
        TSR   #MCBF,R3
 L61:   BTSR  (R3)+,(R2)+
        DEC   R4
        BRZC  L61
        TSR   R2,PDRBF
        DEC   NRPS
        BRZC  L62
        JMS   R2,DWR
 L62:   DEC   C
        BRSC  PL46
        CMP   #0,NRPS
        BRZS  L63
        JMS   R2,DWR
 L63:   TSR   NRPS1,NRPS
        JMP   PL21
; BEGINNING OF SECTION FOR RELATION ' 9'
PL15:   TSR   #BF3+6,PNBF
        TSR   #UNTBF,PNBUF
```

```
        CLR   PKN
        TSR   #7,PKL
PL47:   JMS   R2,PECHEK
        JMS   R2,PKCHEK
        JMS   R2,PNCON
        TSR   #BF3+15,R2
        TSR   #UNTBF+10,R3
        TSR   #7,R4
PL31:   BTSR  (R2)+,(R3)+
        DEC   R4
        BRZC  PL31
        BTSR  BF3+24,COMBF1
        JMS   R2,COMCHK
        BTSR  BF3+24,UNTBF+17
        TSR   FRBF+6,YRUL
        TSR   FRBF+10,YRLL
        BTSR  BF3+25,YLBF
        BTSR  BF3+26,YLBF+1
        JMS   R2,YRLIM
        TSR   #BF3+25,R2
        TSR   #UNTBF+20,R3
        TSR   #7,R4
PL32:   BTSR  (R2)+,(R3)+
        DEC   R4
        BRZC  PL32
        BTSR  BF3+34,YNBF
        JMS   R2,YNCON
        BTSR  YNBF,UNTBF+27
        BTSR  BF3+35,YNBF
        JMS   R2,YNCON
        BTSR  YNBF,UNTBF+30
        TSR   FRBF+12,YRUL
        TSR   FRBF+14,YRLL
        BTSR  BF3+37,YLBF
        BTSR  BF3+40,YLBF+1
        JMS   R2,YRLIM
        TSR   #BF3+36,R2
        TSR   #UNTBF+31,R3
        TSR   #10,R4
PL33:   BTSR  (R2)+,(R3)+
        DEC   R4
        BRZC  PL33
        BCMP  BF3+36,BF3+45
        BRSS  PL54
        TSR   FRBF+16,YRUL
        TSR   FRBF+20,YRLL
        TSR   BF3+46,YLBF
        JMS   R2,YRLIM
        TSR   #BF3+46,R2
        TSR   #UNTBF+41,R3
        TSR   #6,R4
PL34:   BTSR  (R2)+,(R3)+
        DEC   R4
        BRZC  PL34
        BTSR  BF3+54,YNBF
        JMS   R2,YNCON
```

```
        BTSR  YNBF,UNTBF+47
        BTSR  BF3+55,YNBF
        JMS   R2,YNCON
        BTSR  YNBF,UNTBF+50
        TSR   PDRBF,R2
        TSR   #52,R4
        TSR   #UNTBF,R3
L70:    BTSR  (R3)+,(R2)+
        DEC   R4
        BRZC  L70
        TSR   R2,PDRBF
        DEC   NRPS
        BRZC  L71
        JMS   R2,DWR
L71:    DEC   C
        BRSC  PL61
        CMP   #0,NRPS
        BRZS  L72
        JMS   R2,DWR
L72:    TSR   NRPS1,NRPS
        JMP   PL21
PL61:   JMP   PL47
PL54:   SWRITE  N4,RNERR
PL55:   WAITR   N4,PL55
        STCP
        JMP   PL47
RNERR:  WORD  47,0,47
BCI % PRESENT-RANK IS LESS THAN SUBS-RANK %
        BYTE  15,12
        EVEN
UNTBF:  WORD  0
        .=.+100
; BEGINNING OF SECTION FOR RELATION '10'
PL16:   TSR   #BF3+6,PNBF
        TSR   #MINBF,PNBUF
        CLR   PKN
        TSR   #11,PKL
PL52:   JMS   R2,PECHEK
        JMS   R2,PKCHEK
        JMS   R2,PNCON
        BTSR  BF3+15,MINBF+10
        BTSR  BF3+16,MINBF+11
        BTSR  BF3+17,MINBF+12
        TSR   PDRBF,R2
        TSR   #13,R4
        TSR   #MINBF,R3
L73:    BTSR  (R3)+,(R2)+
        DEC   R4
        BRZC  L73
        TSR   R2,PDRBF
        DEC   NRPS
        BRZC  L74
        JMS   R2,DWR
L74:    DEC   C
        BRSC  PL52
        CMP   #0,NRPS
```

```
        BRZS  L75
        JMS   R2,DWR
   L75: TSR   NRPS1,NRPS
        JMP   PL21
MINBF:  WORD  0
        .=.+12
 ; BEGINNING OF SECTION FOR RELATION '11'
PL17:   CLR   PKN
        TSR   #7,PKL
PL107:  JMS   R2,PECHEK
        JMS   R2,PKCHEK
        TSR   #BF3+6,PNBF
        TSR   #REMBF,PNBUF
        JMS   R2,PNCON
        TSR   #BF3+15,PNBF
        TSR   #REMBF+10,PNBUF
        JMS   R2,PNCON
        TSR   FRBF+6,YRUL
        TSR   FRBF+10,YRLL
        TSR   BF3+24,YLBF
        JMS   R2,YRLIM
        TSR   FRBF+12,YRUL
        TSR   FRBF+14,YRLL
        BTSR  BF3+65,YLBF
        BTSR  BF3+66,YLBF+1
        JMS   R2,YRLIM
        TSR   #BF3+24,R2
        TSR   #REMBF+20,R3
        TSR   #50,R4
  PL35: BTSR  (R2)+,(R3)+
        DEC   R4
        BRZC  PL35
        TSR   PDRBF,R2
        TSR   #66,R4
        TSR   #REMBF,R3
   L76: BTSR  (R3)+,(R2)+
        DEC   R4
        BRZC  L76
        TSR   R2,PDRBF
        DEC   NRPS
        BRZC  L77
        JMS   R2,DWR
   L77: DEC   C
        BRSC  PL107
        CMP   #0,NRPS
        BRZS  L100
        JMS   R2,DWR
  L100: TSR   NRPS1,NRPS
        JMP   PL21
REMBF:  WORD  0
        .=.+74
 ; BEGINNING OF SECTION FOR RELATION '12'
PL20:   TSR   #BF3+6,PNBF
        TSR   #DECBF,PNBUF
        CLR   PKN
        TSR   #11,PKL
```

```
PL53:  JMS   R2,PECHEK
       JMS   R2,PKCHEK
       JMS   R2,PNCON
       TSR   FRBF+6,YRUL
       TSR   FRBF+10,YRLL
       BTSR   BF3+17,YLBF
       BTSR   BF3+20,YLBF+1
       JMS   R2,YRLIM
       BTSR   BF3+21,COMBF1
       JMS   R2,COMCHK
       TSR   #BF3+15,R2
       TSR   #DECBF+10,R3
       TSR   #5,R4
PL36:  BTSR   (R2)+,(R3)+
       DEC   R4
       BRZC   PL36
       TSR   PDRBF,R2
       TSR   #16,R4
       TSR   #DECBF,R3
L101:  BTSR   (R3)+,(R2)+
       DEC   R4
       BRZC   L101
       TSR   R2,PDRBF
       DEC   NRPS
       BRZS   L102
       JMS   R2,DWR
L102:  DEC   C
       BRSC   PL53
       CMP   #0,NRPS
       BRZS   L103
       JMS   R2,DWR
L103:  TSR   NRPS1,NRPS
       JMP   PL21
DECBF: WORD   0
       .=.+14
  BF1: WORD   7.
  BF2: WORD   8.
  BF7: WORD   2
ERADDR: WORD   0
  BF3: WORD   120,0,120
       .=.+120
  DIS: WORD   1
  CYL: WORD   55
   BC: WORD   -200
;
       ;
       ; THE SUBROUTINES USED IN THE INPUT VALIDATIONS STAGE
       ;
       ; BEGINNING OF SUBROUTINE PEFORM
       ;
PEFORM: TSR   R2,D
       TSR   #120,BF3
   L1: SREAD   N2,BF3
   L2: WAITR   N2,L2
       TSR   #120,R4
       TSR   #BF3+6,R2
```

```
        TSR   #FBF+60,R3
  L65:  BTSR   (R2)+,(R3)+
        DEC   R4
        BRZC  L65
        SWRITE  N3,FBF
  L44:  WAITR   N3,L44
        TSR   #120,R2
        TSR   #6,R3
        CLR   R6
        CLR   R7
   L4:  CLR   R5
        CLR   R4
        BTSR   BF3(R3),R4
  L21:  BCMP   BLANK,R4
        BRZS   L24
        ADD   #2,R6
        SUB   #60,R4
   L5:  SUB   #1,R4
        BRZS   L25
        ADD#2,R5
        CMP   #10,R5
        BRZC  L5
        SWRITE  N3,BF4
   L7:  WAITR   N3,L7
        JMP   L40
  L24:  JMP   L15
  L25:  JMP   L6(R5)
  BF4:  WORD    14.,0,14.
        BCI. %UNKNOWN FORMAT%
    A:  WORD  1
    C:  WORD  0
    B:  WORD  4,0,4
        .=.+4
BLANK:  WORD  40
        EVEN
 FMT1:  WORD  0
  FMT:  WORD  0
        .=.+200
    D:  WORD 0
  FBF:  WORD  122.,0,122.
BCI %THE FORMAT OF FIELDS OF THIS RELATION IS %
        .=.+120
        EVEN
   L6:  BRN   L10
        BRN   L11
        BRN   L12
        BRN   L13
  L10:  BTSR   #0,FMT1+1
        BRN   L22
  L11:  BTSR   #20,FMT1+1
        BRN   L22
  L12:  BTSR   #40,FMT1+1
        BRN   L22
  L13:  BTSR   #60,FMT1+1
  L22:  BTSR   BF3+1(R3),R4
        SUB   #60,R4
```

```
        BRZC  L45
        BTSR  B+11,R3
        SUB   #60,R3
        ADD   R3,R2
        TSR   R2,C
        DEC   C
        BRSC  L42
        JMP   L40
  L42:  TSR   D,R2
        RTS   R2
 NREF:  WORD  50,0,50
 BCI %  THE NO. OF RECORDS TO BE PLACED IS %
        .=.+4
        EVEN
        ;
        ; END OF NOFREC
        ;
        ;
     ;
     ; BEGINNING OF SUBROUTINE PECHEK
     ;
PECHEK: TSR   R2,D
        TSR   #2,R6
        TSR   FMT,R2
        SUB   #1,R2
        BRSS  PEL3
        TSR   #2,MQ
        MPY   R2,R7
        TSR   FMT+2(R7),BF3
        TSR   FMT+2(R7),BF3+4
 PEL1:  SREAD  N2,BF3
 PEL2:  WAITR  N2,PEL2
        TSR   #6,R3
PEL11:  CLR   R4
        BTSR  FMT+1(R6),R4
        CLR   R5
        BTSR  FMT(R6),R5
        SUB   #1,R4
        BRSS  PEL4
        SUB   #20,R4
        BRSS  PEL5
        SUB   #20,R4
        BRSS  PEL6
        SUB   #20,R4
        BRSS  PEL7
        SWRITE  N3,PEROR
 PEL10: WAITR N3,PEL10
        ADD   #2,R6
        ADD   R5,R3
        BRN   PEL11
 PEL3:  SWRITE  N4,PEERR
PEL12:  WAITR  N4,PEL12
        JMP   PEL36
 PEL5:  JMP   PEL23
 PEL6:  JMP   PEL30
 PEL7:  JMP   PEL35
```

```
PEL4:  CLR   R4
       BTSR  BF3(R3),R4
PEL15: SUB   #60,R4
       BRSS  PEL16
       SUB   #12,R4
       BRSC  PEL16
       ADD   #1,R3
       SUB   #1,R5
       BRZS  PEL17
       BRN   PEL4
PEL16: SWRITE  N3,FMERR
PEL20: WAITR   N3,PEL20
       BRN   PEL22
PEL17: SUB   #1,R2
       BRSS  PEL22
       ADD   #2,R6
       JMP   PEL11
PEL22: TSR   #2,R6
       JMP   PEL36
PEL23: CLR   R4
       BTSR  BF3(R3),R4
PEL25: SUB   #1,R4
       BRSS  PEL27
       SUB   #1,R4
       BRSC  PEL27
       ADD   #1,R3
       SUB   #1,R5
       BRZS  PEL26
       BRN   PEL23
PEL26: JMP   PEL17
PEL27: JMP   PEL16
PEL30: CLR   R4
       BTSR  BF3(R3),R4
       BCMP  BLANK,R4
       BRZS  PEL37
PEL32: SUB   #101,R4
       BRSS  PEL33
       SUB   #32,R4
       BRSC  PEL33
PEL37: ADD   #1,R3
       SUB   #1,R5
       BRZS  PEL34
       BRN   PEL30
PEL33: JMP   PEL16
PEL34: JMP   PEL17
PEL35: ADD   R5,R3
       JMP   PEL17
PEL36: TSR   D,R2
       RTS   R2
PERCR: WORD  42,0,42
       BCI   %  FORMAT CORRUPTED DURING TRANSFER%
PEERR: WORD  42,0,42
       BCI   %  NO OF FIELDS IN RELATION IS ZERO%
FMERR: WORD  40,0,40
       BCI   %  FIELD NOT MATCHING WITH FORMAT%
   ;
```

```
 ; END OF SUBROUTINE PECHEK
 ;
  ;
  ;
  ; SUBROUTINE PNCON
  ;
PNCON: TSR  R2,D
      JMS   R3,PNCV
      JMS   R3,PNDC
      TSR   D,R2
      RTS   R2
PNBUF: WORD  0
 PNBF: WORD  0
  ;
  ; END OF PNCON
  ;
  ;
  ; SUBROUTINE PNCV
  ;
PNCV: TSR   #2,R6
      TSR   #CD,R5
PNL4: TSR   PNBF,R4
      TSR   #2,R2
      TSR   (R5)+,R7
PNL5: BCMP   (R4)+,(R7)+
      BRZS   NXT
      DEC   R6
      BRZS   NOMTCH
      BRN   PNL4
  NXT: DEC   R2
      BRZS   PNL6
      BRN   PNL5
  CD: WORD   CD1,CD2
  CD1: BCI   %IC%
  CD2: BCI   %SS%
NOMTCH: SWRITE   N4,PNBF6
PNL7: WAITR   N4,PNL7
      BRN   PNL6
PNBF6: WORD   14.,0,14.
      BCI   % ERROR IN CODE%
PNL6: RTS   R3
  ;
  ; END OF PNCV
  ;
  ;
  ; SUBROUTINE PNDC
  ;
PNDC: CLR   R4
      CLR   R7
      TSR   #5,R5
      CLR   R6
      TSR   PNBF,R2
      ADD   #2,R2
PNL15: BTSR   (R2)+,R7
      ADD   R7,R4
      SUB   ZERO,R4
```

```
        DEC    R5
        BRZC   PNL15
PNL10:  CMP    R4,TEN
        BRSS   PNL11
PNL12:  SUB    TEN,R4
        ADD    ONE,R6
        BRN    PNL10
PNL11:  ADD    R6,R4
        CLR    R6
        CMP    R4,TEN
        BRSC   PNL12
        ADD    ZERO,R4
        TSR    PNBUF,R2
        BTSR   R4,7(R2)
        TSR    #7,R5
        TSR    PNBF,R6
        TSR    PNBUF,R7
PNL13:BTSR     (R6)+,(R7)+
        DEC    R5
        BRZC   PNL13
        RTS    R3
  TEN:  WORD   10.
  ONE:  WORD   1
 ZERO:  WORD   60
        ;
        ; END OF PNDC
        ;
        ;
        ;
        ; SUBROUTINE YNCON
        ;
YNCON:  BCMP   #131,YNBF
        BRZS   YNL4
        BCMP   #116,YNBF
        BRZS   YNL6
   NM:  SWRITE  N3,YNBF5
YNL20:  WAITR   N3,YNL20
        BRN    YNL1
YNBF5:  WORD   35,0,35
BCI % INCORRECT INPUT, NOT Y OR N %
        EVEN
 YNL4:  BTSR   #377,YNBF
        BRN    YNL1
 YNL6:  CLR    YNBF
 YNL1:  RTS    R2
 YNBF:  BYTE   0
        EVEN
        ;
        ; END OF YNCON
        ;
        ;
        ;
        ; SUBROUTINE YEARLIMITS
        ;
YRLIM:  TSR    #YRUL,R3
        TSR    #YRLL,R4
```

```
        TSR   #YLBF,R5
        BCMP  (R3),(R5)
        BRSS  YLL1
        BCMP  (R3),(R5)
        BRZC  YLL2
        BCMP  1(R3),1(R5)
        BRSS  YLL1
 YLL2:  BCMP  (R5),(R4)
        BRSS  YLL3
        BCMP  (R5),(R4)
        BRZC  YLL4
        BCMP  1(R5),1(R4)
        BRSS  YLL3
        BRN   YLL4
 YLL1:  SWRITE  N4,YLBF1
 YLL5:  WAITR   N4,YLL5
        STOP
        JMP   YLL4
YLBF1:  WORD  44,0,44
BCI %THE YEAR VALUE EXCEEDS UPPER LIMIT%
        BYTE  15,12
        EVEN
 YLL3:  SWRITE  N4,YLBF2
 YLL6:  WAITR   N4,YLL6
        STOP
        JMP   YLL4
YLBF2:  WORD  42,0,42
BCI %THE YEAR VALUE BELOW LOWER LIMIT%
        BYTE  15,12
        EVEN
 YLBF:  WORD  2,0,2
        .=.+2
 YLL4:  RTS   R2
 YRUL:  WORD  0
 YRLL:  WORD  0
        ;
        ; END OF  YEARLIMITS
        ;
        ;
        ;
        ; SUBROUTINE AREA-SELECT
        ; THIS SUBROUTINE SELECTS THE CYLINDER,SURFACE
        ; AND SECTOR ON WHICH THE NEXT RECORD IS
        ; TO BE WRITTEN
        ; IT FIRST CHECKS WHETHER THE SECTOR NO. HAS BECOME
        ; GREATER THAN 10, THEN IT PUTS IT BACK TO 1 AND
        ; INCREMENTS THE SURFACE NO. BY 1 WHEN THIS OCCURS.
        ; NEXT IT CHECKS WHETHER THE SURFACE NO. EXCEEDS
        ; 9, IT THEN INCREMENTS THE CYLINDER NO.
        ; BY 1 AND SETS THE SURFACE NO. TO 0
        ; AND THE SECTOT NO. TO 1.
        ;
ARSLCT: TSR   R2,D
        TSR   DIS,R2
        TSR   #13,R3
PL101:  LSL   R2
```

```
        DEC   R3
        BRZC   PL101
        CMP   #54000,R2
        BRZC   PL100
        TSR   DIS,R2
        TSR   #5,R3
PL102:  ASR   R2
        DEC   R3
        BRZC   PL102
        CMP   #11,R2
        BRZC   PL103
        ADD   #1,CYL
        TSR   #1,DIS
        JMP   PL100
PL103:  INC   R2
        TSR   #5,R3
PL104:  LSL   R2
        DEC   R3
        BRZC   PL104
        INC   R2
        TSR   R2,DIS
PL100:  TSR   D,R2
        RTS   R2
        ;
        ; END OF AREA-SELECT
        ;
        ;
;DISK I/O ROUTINES.
SET:TSR #RER.02,@#200
CLR @#202
TSR #ERR.00,204
CLR @#206
RTS R4
      READ:TSR #4107,-( 1)
BRN  RW
WRITE:TSR #4113,-( 1)
RW:TSB #200,@#177456
BRZS RW
TSR @(R4)+,@#177452
TSR @(R4)+,@#177444
TSR (R4)+,@#177460
        TSR   @(R4)+,@#177462
TSR ( 1)+,@#177454
WAIT
RER.02:CMP(1)+,(1)+
SENSE: TSB   #1,@#177456
        BRZC  SENSE
        TSB   #40000,@#177456
        BRZS  ERR.11
        RTS   R4
ERR.11: STOP
ERR.00:WORD 0
        ;
        ;
        ; SUBROUTINE FOR COMMAND-REGION-CHECK
        ;
```

```
COMCHK: CLR   CMF
        CLR   R3
        BTSR  COMBF1,R3
        BCMP  #116,R3
        BRZS  CML21
        BCMP  #105,R3
        BRZS  CML21
        BCMP  #127,R3
        BCMP  #123,R3
        BRZS  CML21
        BCMP  #103,R3
        BRZS  CML21
        SWRITE  N4,CMERR
CML22:  WAITR   N4,CML22
        TSR   #1,CMF
 CML21: RTS   R2
CMBF1:  WORD  0
CMERR:  WORD  47,0,47
BCI % THE COMMAND REGION IS GIVEN WRONGLY %
         BYTE  15,12
        EVEN
  CMF:  WORD  0
 COMBF1: WORD  0
        ;
        ; END OF COMCHK
        ;
        ;
;
        ;
    ; SUBROUTINE FOR DISK -WRITE
        ;
  DWR:  JMS   R4,WRITE
        WORD  CYL,DIS,DRBF,BC
        TSR   #DRBF,PDRBF
        TSR   NRPS1,NRPS
        INC   DIS
        JMS   R2,ARSLCT
        STOP
        RTS   R2
        ;
        ;
        ; BEGINNING OF SUBROUTINE PKCHEK
        ;
PKCHEK: TSR   PKN,R3
        TSR   PKB,R4
        CMP   #0,PKN
        BRZS  PKL40
 PKL1:  TSR   PKL,R5
        TSR   #BF3+6,R6
 PKL2:  BCMP  (R4)+,(R6)+
        BRZC  PKL3
        DEC   R5
        BRZC  PKL2
        SWRITE  N4,PKERR
 PKL4:  WAITR   N4,PKL4
        SWRITE  N4,BF3
```

```
 PKL5: WAITR   N4,PKL5
       STOP
       JMP   PKL41
PKL40: JMP   PKL6
PKERR: WORD   40,0,40
       BYTE   15,12
       BCI % DUPLICATION OF PRIMARY KEY %
       BYTE   15,12
       EVEN
 PKL3: ADD   R5,R4
       DEC   R3
       BRZS   PKL6
       JMP   PKL1
 PKL6: TSR   PKL,R5
       TSR   #BF3+6,R6
 PKL7: BTSR   (R6)+,(R4)+
       DEC   R5
       BRZC   PKL7
       INC   PKN
PKL41: RTS   R2
  PKN: WORD   0
  PKL: WORD   0
  PKB: WORD   0
       ;
       ; END OF PKCHEK
       ;
       ;
       ;
       ; SUBROUTINE COMCOD
       ;
       ;
COMCOD: TSR   R2,D
       TSR   CMBF2,R2
       TSR   CMBF3,R3
       TSR   #2,R7
       TSR   #22,R4
 CML2: CLR   R6
 CML1: CLR   R5
       BTSR   (R2)+,R5
       SUB   #60,R5
       ADD   R5,R6
       DEC   R4
       BRZC   CML1
       DEC   R7
       BRZS   CML3
       BTSR   R6,(R3)+
       TSR   #11,R4
       BTSR   R6,TEMP2
       JMP   CML2
 CML3: TSR   #6,R4
       TSR   CMBF2,R7
       ADD   #37,R7
 CML4: CLR   R5
       BTSR   (R7)+,R5
       SUB   #60,R5
       ADD   R5,R6
```

```
        DEC    R4
        BRZC   CML4
        BTSR   R6,(R3)+
        CLR    R5
        BTSR   -4(R3),R5
        ADD    R5,R6
        BTSR   R6,(R3)
        TSR    D,R2
        RTS    R2
CMBF2:  WORD   0
CMBF3:  WORD   0
TEMP2:  WORD   0
COMBF:  WORD   0
        .=.+60
        ;
        ; END OF COMCOD
        ;
        ;
        ;
        ; SUBROUTINE PROF-QUAL-GRADE-CHECKING
    ; THIS SUBROUTINE CHECKS FOR THE ALLOWABLE RANGE OF GRADES
    ; THE ONLY ALLOWED GRADES ARE A,B,C & D
        ;
        ;
PQLCHK: CLR    R3
        CLR    R4
        CLR    PQF
        BTSR   PQLBF,R3
        BCMP   #104,R3
        BRZS   PQL6
        BCMP   #101,R3
        BRZS   PQL6
        BCMP   #102,R3
        BRZS   PQL6
        BCMP   #103,R3
        BRZS   PQL6
        SWRITE  N4,PQERR
 PQL5:  WAITR   N4,PQL5
        TSR    #1,PQF
        JMP    PQL6
PQERR:  WORD   37,0,37
        BYTE   15,12
        BCI    % THE GRADING CODE IS WRONG %
        BYTE   15,12
        EVEN
 PQBF:  WORD   15,0,15
        .=.+16
 PQL6:  RTS    R2
PQLBF:  WORD   0
  PQF:  WORD   0
        ;
      ; END OF PQLCHK
        ;
        ;
        ;
        ; SUBROUTINE ARMCON
```

```
        ;
ARMCON: CLR   R3
        BTSR  ARMBF1,R3
        SUB   #60,R3
        TSR   #12,MQ
        MPY   R3,R3
        TSR   MQ,R3
        CLR   R4
        BTSR  ARMBF1+1,R4
        SUB   #60,R4
        ADD   R4,R3
        BTSR  R3,ARMBF1
        RTS   R2
ARMBF1: WORD  0
ARMBF:  WORD  14,0,14
        .=.+14
        ;
        ; END OF ARMCON
        ;
        ;
        ;
        ; SUBROUTINE FOR FOREIGN-LANGUAGE CHECKING AND TRANSFERRING
        ; IT FIRST CHECKS WHETHER THE LEVEL OF EXAM PASSED IS
        ; WITHIN RANGE OR NOT , THE ONLY ALLOWED ONES BEING 1,2 & 3
        ; IT ALSO COMPUTES THE POSITIONAL CHECKSUM AND APPENDS
        ; IT TO THE END OF THE RECORD.
        ; IT ALSO TRANSFERS THE DATA FROM THE READER BUFFER
        ; TO THE FOREIGN LANGUAGE BUFFER
        ;
 FLCON: CLR   R3
        CLR   R5
        BTSR  BF3+17,R3
        BTSR  R3,FLBF+12
        SUB   #61,R3
        BRSS  FLL1
        ADD   R3,R5
        INC   R5
        SUB   #3,R3
        BRSC  FLL1
        CLR   R3
        BTSR  BF3+15,R3
        BTSR  R3,FLBF+10
        SUB   #60,R3
        TSR   #12,MQ
        MPY   R3,R3
        TSR   MQ,R3
        CLR   R4
        BTSR  BF3+16,R4
        BTSR  R4,FLBF+11
        SUB   #60,R4
        ADD   R4,R3
        CLR   R4
        BTSR  BF3+20,R4
        BTSR  R4,FLBF+13
        SUB   #60,R4
        TSR   #12,MQ
```

```
        MPY   R4,R4
        TSR   MQ,R4
        ADD   R4,R3
        CLR   R4
        BTSR  BF3+21,R4
        BTSR  R4,FLBF+14
        SUB   #60,R4
        ADD   R4,R3
        ADD   R5,R3
        BTSR  R3,FLBF+15
        BRN   FLL2
 FLBF:  WORD  15,0,15
        .=.+16
FLERR:  WORD  45,0,45
        BCI   % HIGHEST EXAM PASSED IS OUT OF RANGE %
        EVEN
 FLL1:  SWRITE  N4,FLERR
 FLL3:  WAITR   N4,FLL3
        STOP
 FLL2:  RTS   R2
        ;
        ; END OF FLCON
        ;
        ;
        ;
        ; SUBROUTINE MEDCOD
 ; FROM INPUT BUFFER AS PLACED BY THE READER
 ; TO THE MEDICALBUFFER . IT ALSO COMPUTES
 ; THE DIGITWISE CHECKSUMS OF EACH OF THE BLOCKS
 ; S,H,A,P,E SEPARATELY AS WELL AS THE CHECKSUMS
 ; OF THE FIELDS P,M,Y,V .
 ; IT APPENDS THE CHECKSUM OF THE BLOCKS AT
 ; THE END OF EACH BLOCK AND THE CHECKSUM OF
 ; EACH FIELD AT THE END OF THE RECORD ITSELF
 ; R3 CONTAINS THE ADDRESS OF THE INPUT STRING
 ; IN THE READER BUFFER AND R6 CONTAINS THE ADDRESS THE OF THE POS
 ; OF THE POSITION IN MEDICALBUFFER WHERE THE
 ; THE STRING IS TO BE MOVED
 ; TEMP CONTAINS THE VALUE OF THE CHECKSUMMFOR
 ; EACH BLOCK AND APPENDS IT TO THE END OF EACH
 ; BLOCK AS IT IS PASSED,
 ; P CONTAINS THE CHECKSUM OF THE PRIOD FIELD,
 ; M CONTAINS THE CHECKSUM OF THE MONTH-FIELD,
 ; Y CONTAINS THE CHECKSUM OF THE YEAR FIELD,
 ; V CONTAINS THE CHECKSUM OF VALUE FIELD9
        ;
MEDCOD: TSR   R2,D
        TSR   #5,R5
        CLR   R2
        CLR   P
        CLR   M
        CLR   Y
        CLR   V
        CLR   TEMP
        TSR   #BF3+15,R3
        TSR   #MCBF+10,R6
```

```
MCL2:  CLR   R4
       CLR   R7
       BTSR  (R3)+,R4
       BTSR  R4,(R6)+
       SUB   #60,R4
       BTSR  (R3)+,R7
       BTSR  R7,(R6)+
       SUB   #60,R7
       ADD   R7,R4
       JMP   MCL3(R2)
MCL3:  ADD   R4,P
       ADD   R4,TEMP
       ADD   #20,R2
       JMP   MCL2
       ADD   R4,M
       ADD   R4,TEMP
       ADD   #20,R2
       JMP   MCL2
       ADD   R4,Y
       ADD   R4,TEMP
       CLR   R4
       BTSR  (R3)+,R4
       BTSR  R4,(R6)+
       SUB   #60,R4
       ADD   R4,TEMP
       ADD   R4,V
       BTSR  TEMP,(R6)+
       CLR   TEMP
       CLR   R2
       DEC   R5
       BRZC  MCL2
       BTSR  P,(R6)+
       BTSR  M,(R6)+
       BTSR  Y,(R6)+
       BTSR  V,(R6)+
       JMP   MCL1
    P: WORD  0
    M: WORD  0
    Y: WORD  0
    V: WORD  0
MCBF:  WORD  100,0,100
       .=.+100
TEMP:  WORD  0
MCL1:  TSR   D,R2
       RTS   R2
       ;
       ;
       ; SUBROUTINE TO VALIDATE INSTITUTE OF COMMISSIONING
       ;
INCVAL: TSR  #2,R3
       TSR   #INCD,R4
INCL3: TSR   CMBUF,R5
       TSR   #3,R7
       TSR   (R4)+,R6
INCL5: BCMP  (R6)+,(R5)+
       BRZS  INCL1
```

```
        SUB    #1,R3
        BRZS   INCL2
        BRN    INCL3
INCL1:  SUB    #1,R7
        BRZS   INCL4
        BRN    INCL5
INCD:   WORD   INC1,INC2
INC1:   BCI    %IMA%
INC2:   BCI    %OTS%
INCL2:  SWRITE  N4,INCBF
INCL6:  WAITR   N4,INCL6
        STOP
        BRN    INCL4
INCBF:  WORD   40,0,40
        BYTE   15,12
        BCI    % INST. OF COMM. GIVEN WRONGLY %
        EVEN
INCL4:  RTS    R2
CMBUF:  WORD   0
        ;
        ; END OF INCVAL
        ;
        ;
        ;
        ;SUBROUTINE QUALCODE
        ;
QALCOD: TSR    R2,D
        CLR    QLTOT
        CLR    QLTOT1
        TSR    #2,R7
        TSR    #QULBF+10,R6
        TSR    QLBF,R2
QLL2:   TSR    #2,R5
QLL1:   CLR    R4
        CLR    R3
        BTSR   (R2)+,R4
        BTSR   R4,(R6)+
        SUB    #60,R4
        ADD    R4,QLTOT1
        TSR    #12,MQ
        MPY    R4,R4
        TSR    MQ,R4
        BTSR   (R2)+,R3
        BTSR   R3,(R6)+
        SUB    #60,R3
        ADD    R3,QLTOT1
        ADD    R3,R4
        ADD    R4,QLTOT
        SUB    #1,R5
        BRZC   QLL1
        SUB    #1,R7
        BRZS   QLL3
        CLR    R3
        BTSR   (R2)+,R3
        BTSR   R3,(R6)+
        SUB    #60,R3
```

```
        ADD R3,QLTOT1
        ADD  R3,QLTOT
        BRN  QLL2
QLTOT: WORD  0
QLTOT1: WORD  0
        ;
 QLL3: BTSR  QLTOT,(R6)+
        BTSR  QLTOT1,(R6)+
        TSR  D,R2
        RTS  R2
 QULBF: WORD  0
        .=.+30
 QLBF: WORD  0
        ;
        ; END OF QUALCODE
        ;
        ;
  L40: NOP
        ; THIS COMPLETES THE INPUT SECTION
        END

;
        ;
        ;
        ; THE SUBROUTINES USED IN THE OTHER STAGES OF DATA VALIDATION
        ; THESE ROUTINES ARE COMPLIMENTARY ROUTINES TO THE SUBROUTINES USED IN T
        ; INPUT STAGE AND ARE TO BE USED IN THE RETRIEVAL OF THE INFORMATION FRO
        ; THE RELATIONS
        ; THESE SUBROUTINES WERE TESTED UNDER SEPARATION SO THEY MAY NEED SLIGHT
        ; RECONDITIONING AS TO POSITION AND VARIABLES AND LABELS
        ;
        ;
        ; SUBROUTINE BIN-TO-ASCII
BINASC: CLR  R3
        TSR  BABUF,R5
        CLR  MQ
        BTSR  BABF,MQ
        TSR  #144,R4
        DIV  R4,R3
        BCMP  #0,MQ
        BRZS  BAL1
        ADD  #60,MQ
        BTSR  MQ,(R5)+
        JMP  BAL3
 BAL1: BTSR  #60,(R5)+
 BAL3: TSR  R3,MQ
        CLR  R3
        TSR  #12,R4
        DIV  R4,R3
        BCMP  #0,MQ
        BRZS  BAL2
        ADD  #60,MQ
        BTSR  MQ,(R5)+
        JMP  BAL4
 BAL2: BTSR  #60,(R5)+
 BAL4: ADD  #60,R3
```

```
        BTSR   R3,(R5)+
        RTS   R2
 BABF:  BYTE   0
        EVEN
BABUF:  WORD   0
        ;
        ; END OF BIN-TO-ASCII
        ;
        ;
       ;
        ;
        ; END OF MEDCOD
        ;
        ;
        ; SUBROUTINE MEDVAL
        ;
;
  ; THIS SUBROUTINE VALIDATES THE BLOCKS AND FIELDS OF
  ; THE MEDICAL FILE OF OFFICERS BY VALIDATING
  ; THE CHECKSUMS CALCULATED BY THE MEDCOD ROUTINE
 ; OF THE BLOCKS AS WELL AS THE CHECKSUMS OF
 ; AND APPENDED TO THEB POSIYIONS AT THE END
        ; EACH TYPE OF FIELD AT THE END OF THE RECORD
        ;
        ;
MEDVAL: TSR   R2,D
        CLR   R2
        CLR   R7
        CLR   TEMP1
        CLR   SS
        CLR   HH
        CLR   AA
        CLR   PP
        CLR   EE
        CLR   SF
        CLR   HF
        CLR   AF
        CLR   PF
        CLR   EF
        CLR   PRF
        CLR   P1
        CLR   M1
        CLR   Y1
        CLR   V1
        TSR   #20,R6
        TSR   #MCBF+16,R2
 MVL3:  CLR   R3
 MVL1:  CLR   R4
        CLR   R5
        BTSR   (R2)+,R4
        SUB   #60,R4
        BTSR   (R2)+,R5
        SUB   #60,R5
        ADD   R5,R4
        JMP   MVL2(R3)
 MVL2:  ADD   R4,P1
```

```
        ADD     R4,TEMP1
        ADD     #20,R3
        JMP     MVL1
        ADD     R4,M1
        ADD     R4,TEMP1
        ADD     #20,R3
        JMP     MVL1
        ADD     R4,Y1
        ADD     R4,TEMP1
        CLR     R4
        BTSR    (R2)+,R4
        SUB     #60,R4
        ADD     R4,V1
        ADD     R4,TEMP1
        CLR     R5
        BTSR    (R2)+,R5
        BCMP    TEMP1,R5
        BRZC    ML70
MVL26:  CLR     TEMP1
        SUB     #4,R6
        BRSC    MVL3
        CLR     R4
        BTSR    (R2)+,R4
        BCMP    P1,R4
        BRZC    ML53
MVL40:  CLR     R4
        BTSR    (R2)+,R4
        BCMP    M1,R4
        BRZC    ML52
MVL52:  CLR     R4
        BTSR    (R2)+,R4
        BCMP    Y1,R4
        BRZC    ML57
MVL64:  CLR     R4
        BTSR    (R2)+,R4
        BCMP    V1,R4
        BRZC    ML67
        JMP     MVL11
ML53:   JMP     MVL4
ML52:   JMP     MVL5
ML70:   JMP     MVL21(R6)
ML67:   JMP     MVL7
ML57:   JMP     MVL6
ML73:   JMP     MVL47
ML71:   JMP     MVL43
ML72:   JMP     MVL45
MVL21:  JMP     MVL22
        JMP     MVL23
        JMP     MVL24
        JMP     MVL25
        TSR     TEMP1,SS
        TSR     #1,SF
        JMP     MVL26
MVL22:  TSR     TEMP1,EE
        TSR     #1,EF
        JMP     MVL26
```

```
MVL23:  TSR   TEMP1,PP
        TSR   #1,PF
        JMP   MVL26
MVL24:  TSR   TEMP1,AA
        TSR   #1,AF
        JMP   MVL26
MVL25:  TSR   TEMP1,HH
        TSR   #1,HF
        JMP   MVL26
ML65:   JMP   MVL61
ML77;   JMP   MVL57
ML76:   JMP   MVL57
ML50:   JMP   MVL35
ML74:   JMP   MVL51
ML75:   JMP   MVL53
ML54:   JMP   MVL41
ML55:   JMP   MVL37
MVL4:   CMP   #1,SF
        BRZS  MVL27
MVL30:  CMP   #1,HF
        BRZS  MVL31
MVL32:  CMP   #1,AF
        BRZS  MVL33
MVL34:  CMP   #1,PF
        BRZS  ML50
MVL36:  CMP   #1,EF
        BRZS  ML55
        BTSR  P1,-2(R2)
        JMP   MVL40
MVL5:   CMP   #1,SF
        BRZS  ML54
MVL42:  CMP   #1,HF
        BRZS  ML71
MVL44:  CMP   #1,AF
        BRZS  ML72
MVL46:  CMP   #1,PF
        BRZS  ML73
MVL50:  CMP   #1,EF
        BRZS  ML74
        BTSR  M1,-2(R2)
        JMP   MVL52
MVL6:   CMP   #1,SF
        BRZS  ML75
MVL54:  CMP   #1,HF
        BRZS  ML76
MVL56:  CMP   #1,AF
        BRZS  ML77
MVL60:  CMP   #1,PF
        BRZS  ML65
MVL62:  CMP   #1,EF
        BRZS  ML66
        BTSR  Y1,-2(R2)
        JMP   MVL64
MVL7:   CMP   #1,SF
        BRZS  ML60
MVL66:  CMP   #1,HF
```

```
        BRZS  ML61
MVL70:  CMP   #1,AF
        BRZS  ML62
MVL72:  CMP   #1,PF
        BRZS  ML63
MVL74:  CMP   #1,EF
        BRZS  ML64
        BTSR  V1,-2(R2)
        JMP   MVL11
MVL27:  SWRITE  N3,MVSP
  ML1:  WAITR  N3,ML1
        TSR   #1,PRF
        JMP   MVL30
 ML62:  JMP   MVL71
 ML61:  JMP   MVL67
 ML60:  JMP   MVL65
 ML63:  JMP MVL73
 ML64:  JMP   MVL75
 ML66:  JMP   MVL63
MVL31:  SWRITE  N3,MVHP
  ML2:  WAITR  N3,ML2
        TSR   #1,PRF
        JMP   MVL32
MVL33:  SWRITE  N3,MVAP
  ML3:  WAITR  N3,ML3
        TSR   #1,PRF
        JMP   MVL34
MVL35:  SWRITE  N3,MVPP
  ML4:  WAITR  N3,ML4
        TSR   #1,PRF
        JMP   MVL36
MVL37:  SWRITE  N3,MVEP
  ML5:  WAITR  N3,ML5
        TSR   #1,PRF
        JMP   MVL40
MVL41:  SWRITE  N3,MVSM
  ML6:  WAITR  N3,ML6
        TSR   #1,PRF
        JMP   MVL42
MVL43:  SWRITE  N3,MVHM
  ML7:  WAITR  N3,ML7
        TSR   #1,PRF
        JMP   MVL44
MVL45:  SWRITE  N3,MVAM
 ML10:  WAITR  N3,ML10
        TSR   #1,PRF
        JMP   MVL46
MVL47:  SWRITE  N3,MVPM
 ML11:  WAITR  N3,ML11
        TSR   #1,PRF
        JMP   MVL50
MVL51:  SWRITE  N3,MVEM
 ML12:  WAITR  N3,ML12
        TSR   #1,PRF
        JMP   MVL52
MVL53:  SWRITE  N3,MVSY
```

```
 ML13: WAITR   N3,ML13
       TSR    #1,PRF
       JMP    MVL54
MVL55: SWRITE  N3,MVHY
 ML14: WAITR   N3,ML14
       TSR    #1,PRF
       JMP    MVL56
MVL57: SWRITE  N3,MVAY
 ML15: WAITR   N3,ML15
       TSR    #1,PRF
       JMP    MVL60
MVL61: SWRITE  N3,MVPY
 ML16: WAITR   N3,ML16
       TSR    #1,PRF
       JMP    MVL62
MVL63: SWRITE  N3,MVEY
 ML17: WAITR   N3,ML17
       TSR    #1,PRF
       JMP    MVL64
MVL65: SWRITE  N3,MVSV
 ML20: WAITR   N3,ML20
       TSR    #1,PRF
       JMP    MVL66
MVL67: SWRITE  N3,MVHV
 ML21: WAITR   N3,ML21
       TSR    #1,PRF
       JMP    MVL70
MVL71: SWRITE  N3,MVAV
 ML22: WAITR   N3,ML22
       TSR    #1,PRF
       JMP    MVL72
MVL73: SWRITE  N3,MVPV
 ML23: WAITR   N3,ML23
       TSR    #1,PRF
       JMP    MVL74
MVL75: SWRITE  N3,MVEV
 ML24: WAITR   N3,ML24
       TSR    #1,PRF
       JMP    MVL11
MVL11: CMP    #1,PRF
       BRZC   MVL76
       SWRITE  N3,MCBF
MVL77: WAITR   N3,MVL77
MVL76: TSR    D,R2
       RTS    R2
   SS: WORD   0
   HH: WORD   0
   AA: WORD   0
   PP: WORD   0
   EE: WORD   0
   SF: WORD   0
   HF: WORD   0
   AF: WORD   0
   PF: WORD   0
   EF: WORD   0
  PRF: WORD   0
```

```
TEMP1: WORD  0
MVSP:  WORD  31,0,31
       BCI   %S-BLOCK,P-VALUE CORRUPTED%
       EVEN
MVHP:  WORD  31,0,31
       BCI   %H-BLOCK,P-VALUE CORRUPTED%
       EVEN
MVAP:  WORD  31,0,31
       BCI   %A-BLOCK,P-VALUE CORRUPTED%
       EVEN
MVPP:  WORD  31,0,31
       BCI   %P-BLOCK,P-VALUE CORRUPTED%
       EVEN
MVEP:  WORD  31,0,31
       BCI   %E-BLOCK,P-VALUE CORRUPTED%
       EVEN
MVSM:  WORD  31,0,31
       BCI   %S-BLOCK,M-VALUE CORRUPTED%
       EVEN
MVHM:  WORD  31,0,31
       BCI   %H-BLOCK,M-VALUE CORRUPTED%
       EVEN
MVAM:  WORD  31,0,31
       BCI   %A-BLOCK,M-VALUE CORRUPTED%
       EVEN
MVPM:  WORD  31,0,31
       BCI   %P-BLOCK,M-VALUE CORRUPTED%
       EVEN
MVEM:  WORD  31,0,31
       BCI   %E-BLOCK,M-VALUE CORRUPTED%
       EVEN
MVSY:  WORD  31,0,31
       BCI   %S-BLOCK,Y-VALUE CORRUPTED%
       EVEN
MVHY:  WORD  31,0,31
       BCI   %H-BLOCK,Y-VALUE CORRUPTED%
       EVEN
MVAY:  WORD  31,0,31
       BCI   %A-BLOCK,Y-VALUE CORRUPTED%
       EVEN
MVPY:  WORD  31,0,31
       BCI   %P-BLOCK,Y-VALUE CORRUPTED%
       EVEN
MVEY:  WORD  31,0,31
       BCI   %E-BLOCK,Y-VALUE CORRUPTED%
       EVEN
MVSV:  WORD  31,0,31
       BCI   %S-BLOCK,V-VALUE CORRUPTED%
       EVEN
MVHV:  WORD  31,0,31
       BCI   %H-BLOCK,V-VALUE CORRUPTED%
       EVEN
MVAV:  WORD  31,0,31
       BCI   %A-BLOCK,V-VALUE CORRUPTED%
       EVEN
MVPV:  WORD  31,0,31
```

```
        BCI   %P-BLOCK,V-VALUE CORRUPTED%
        EVEN
 MVEV:  WORD  31,0,31
        BCI   %E-BLOCK,V-VALUE CORRUPTED%
        EVEN
   Y1:  WORD  0
   V1:  WORD  0
   P1:  WORD  0
   M1:  WORD  0
        ;
        ; END OF MEDVAL
        ;
        ;
        ; END OF MEDVAL
        ;
        ;
        ; SUBROUTINE FLDEC
        ;
FLDEC:  TSR   R2,D
        CLR   R3
        CLR   R5
        CLR   R4
        TSR   #2,R6
        TSR   #FLBF+16,R3
 FLL4:  BTSR  (R3)+,R4
        SUB   #60,R4
        TSR   #12,MQ
        MPY   R4,R4
        TSR   MQ,R4
        BTSR  (R3)+,R5
        SUB   #60,R5
        ADD   R5,R4
        DEC   R6
        BRZS  FLL5
        STOP
        CLR   R5
        BTSR  (R3)+,R5
        SUB   #60,R5
        ADD   R5,R4
        CLR   R5
        JMP   FLL4
 FLER:  WORD  20,0,20
        BCI   % DATA CORRUPTED %
 FLL5:  CLR   R5
        BTSR  (R3),R5
        STOP
        BCMP  R5,R4
        BRZS  FLL6
        SWRITE  N3,FLER
 FLL7:  WAITR   N3,FLL7
        SWRITE  N3,FLBF
FLL10:  WAITR   N3,FLL10
 FLL6:  RTS   R2
        ;
        ; END OF FLDEC
        ;
```

```
        ;
        ; SUBROUTINE QUALVAL
        ;
GALVAL: TSR   R2,D
        CLR   QLVAL
        TSR   #QULBF+16,R3
        TSR   #2,R6
 QLL4:  TSR   #2,R7
 QLL5:  CLR   R4
        CLR   R5
        BTSR  (R3)+,R4
        SUB   #60,R4
        ADD   R4,R2
        TSR   #12,MQ
        MPY   R4,R4
        TSR   MQ,R4
        BTSR  (R3)+,R5
        SUB   #60,R5
        ADD   R5,R2
        ADD   R5,R4
        ADD   R4,QLVAL
        SUB   #1,R7
        BRZC  QLL5
        STOP
        SUB   #1,R6
        BRZS  QLL6
        CLR   R4
        BTSR  (R3)+,R4
        SUB   #60,R4
        ADD   R4,R2
        ADD   R4,QLVAL
        STOP
        JMP   QLL4
 QLVAL: WORD  0
 QLL6:  CLR   R4
        BTSR  (R3)+,R4
        BCMP  R4,QLVAL
        BRZS  QLL7
        CLR   R4
        BTSR  (R3),R4
        BCMP  R2,R4
        BRZS  QLL12
        SWRITE N3,QLERR
 QLL10: WAITR  N3,QLL10
        BTSR  QULBF+25,BABF
        TSR   #QULBF+25,BABUF
        JMS   R2,BINASC
        BTSR  R4,BABF
        TSR   #QULBF+30,BABF
        JMS   R2,BINASC
        SWRITE N3,QULBF
 QLL11: WAITR  N3,QLL11
        BRN   QLL7
 QLL12: BTSR  QLVAL,QULBF+25
        BRN   QLL13
 QLL7:  BTSR  R2,(R3)
```

```
        RTS   R2
QLERR:  WORD  20,0,20
        BCI   % DATA CORRUPTED %
    D:  WORD  0
        ;
        ; END OF QUALVAL
        ;
        ;
        ; SUBROUTINE YNDECONVERT
YNDECC: TSR   R2,D
        TSR   #10,R3
        CLR   R4
YNL12:  BTSR  A,YNBF
        BRZS  YNL13
        ADD   #1,R4
YNL13:  DEC   R3
        BRZS  YNL14
        BROL  YNBF
        BRN   YNL12
YNL14:  SUB   #5,R4
        BRSS  YNL15
        BTSR  #'Y,YNBF
        BRN   YNL16
YNL15:  BTSR  #'N,YNBF
YNL16:  BTSR  YNBF,YNBF3+6
        SWRITE  N3,YNBF3
YNL17:  WAITR  N3,YNL17
        RTS   R2
    A:  BYTE  200
        EVEN
    D:  WORD  0
        ;
        ; END OF YNDECONVERT
        ;
COMDEC: TSR   R2,D
        CLR   CMDF
        CLR   CMMF
        CLR   TEMP3
        TSR   CMBF4,R2
        TSR   CMBF5,R3
        TSR   #2,R7
        TSR   #22,R4
 CML5:  CLR   R6
 CML6:  CLR   R5
        BTSR  (R2)+,R5
        SUB   #60,R5
        ADD   R5,R6
        DEC   R4
        BRZC  CML6
        STOP
        DEC   R7
        BRZS  CML7
        BTSR  R6,TEMP3
        TSR   #11,R4
        JMP   CML5
 CML7:  TSR   #6,R4
```

```
        TSR    CMBF4,R2
        ADD    #37,R2
CML10:  CLR    R5
        BTSR   (R2)+,R5
        SUB    #60,R5
        ADD    R5,R6
        DEC    R4
        BRZC   CML10
        CLR    R7
        BTSR   (R3)+,R7
        BCMP   TEMP3,R7
        BRZS   CML11
        TSR    #1,CMDF
CML11:  CLR    R7
        BTSR   (R3)+,R7
        BCMP   R6,R7
        BRZS   CML12
        TSR    #1,CMMF
CML12:  CLR    R7
        BTSR   (R3),R7
        ADD    TEMP3,R6
        BCMP   R6,R7
        BRZC   CML13
        JMP    CML14
CML13:  CMP    #1,CMDF
        BRZS   CML15
CML16:  CMP    #1,CMMF
        BRZS   CML17
        BTSR   R6,(R3)
        JMP    CML20
CML14:  BTSR   TEMP3,-4(R3)
        SUB    TEMP3,R6
        BTSR   R6,-2(R3)
        JMP    CML20
CML15:  SWRITE   N3,CDBF1
CML21:  WAITR   N3,CML21
        SWRITE   N3,COMBF
CML22:  WAITR   N3,CML22
        JMP    CML16
CML17:  SWRITE   N3,CDBF2
CML23:  WAITR   N3,CML23
        SWRITE   N3,COMBF
CML24:  WAITR   N3,CML24
CML20:  TSR    D,R2
        RTS    R2
 CMDF:  WORD   0
 CMMF:  WORD   0
TEMP3:  WORD   0
CDBF1:  WORD   26,0,26
        BCI    % ERROR IN DATE-FIELDS %
CDBF2:  WORD   50,0,50
 BCI % ERROR IN PCS-OR-JCO-NCC-RN-CORPS FIELD %
BCI %ERROR IN PCS-OR-JC-NCC-RN-CORPS FIELD%
        EVEN
CMBF4:  WORD   0
CMBF5:  WORD   0
```

```
CCMBF: WORD  62,0,62
       .=.+60
       BYTE  15,12
       EVEN
       ;
```